

Android 专业人士笔记

Android Notes for Professionals



Chapter 91: Menu

Parameter
`inflate(int resources, Menu menu)`
`getMenuInflater()`
Description
Inflate a menu hierarchy from the specified XML resource.
Returns a `MenuItemLister` with this context.
Initializes the contents of the Activity's standard options menu.
You should place your menu items in to menu.
This method is called whenever an item in your options menu is selected.

Section 91.1: Options menu with dividers

In Android there is a default options menu, which can take a number of options. If a larger number of options needs to be displayed, then it makes sense to group those options in order to maintain clarity. Options can be grouped by putting dividers (i.e. horizontal lines) between them. In order to allow for dividers, the following theme can be used:

```
1. Create your custom view skeleton: this is basically the same for every custom view.  
sketches for a custom view that can draw a smiley, called SmileyView.  
  
public class SmileyView extends View {  
    private Paint mPaint;  
    private Paint mSmileyPaint;  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius = new RectF();  
  
    public SmileyView(Context context) {  
        super(context, null, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        super(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
    }  
  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)  
    {  
        // Initialize your points: the false objects are the circles of your virtual  
        // objects are rendered (e.g. color, fill and stroke style, etc). Here we are  
        // for the circle and one black stroke point for the eyes and the mouth.  
        // and then in the Activity:  
        // and finally from the onActivityResult get the payment response  
    }  
  
    private void initPaints() {  
        mSmileyPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        mSmileyPaint.setAntiAlias(true);  
        mSmileyPaint.setColor(Color.YELLOW);  
        mSmileyPaint.setStrokeWidth(2);  
        mSmileyPaint.setStyle(Paint.Style.STROKE);  
        mSmileyPaint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));  
        mSmileyPaint.setAlpha(100);  
        mSmileyPaint.setDither(true);  
    }  
  
    3. Implement your own onMeasure(...) method: this is required so that the parent layouts (A8  
        and A9) can measure the view correctly.  
  
    Android Notes for Professionals
```

Chapter 159: Android PayPal Gateway Integration

Section 159.1: Setup PayPal in your android code

1) First go through PayPal Developer web site and create an application.
2) Now open your manifest file and give the below permissions:
`<uses-permission android:name="android.permission.INTERNET" />`
`<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />`
3) And some required Activity and Services

```
service  
    android:name=".PaypalService"  
    android:exported="false"/>  
    private Paint mPaypalPaint;  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius = new RectF();  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        super(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
    }  
  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)  
    {  
        // Initialize your points: the false objects are the circles of your virtual  
        // objects are rendered (e.g. color, fill and stroke style, etc). Here we are  
        // for the circle and one black stroke point for the eyes and the mouth.  
        // and then in the Activity:  
        // and finally from the onActivityResult get the payment response  
    }  
  
    private void initPaints() {  
        mPaypalPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        mPaypalPaint.setAntiAlias(true);  
        mPaypalPaint.setColor(Color.BLUE);  
        mPaypalPaint.setStrokeWidth(2);  
        mPaypalPaint.setStyle(Paint.Style.STROKE);  
        mPaypalPaint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));  
        mPaypalPaint.setAlpha(100);  
        mPaypalPaint.setDither(true);  
    }  
  
    3. Implement your own onMeasure(...) method: this is required so that the parent layouts (A8  
        and A9) can measure the view correctly.  
  
    Android Notes for Professionals
```

Chapter 28: Creating Custom Views

Section 28.1: Creating Custom Views

If you need a completely customized view, you'll need to subclass `View` (the superclass of all `View`s).

provide your custom `onCreateContextMenu(...)` and drawing `onDraw(...)` methods:

1. Create your custom view skeleton: this is basically the same for every custom view.

sketches for a custom view that can draw a smiley, called SmileyView.

```
public class SmileyView extends View {  
    private Paint mPaint;  
    private Paint mSmileyPaint;  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius = new RectF();  
  
    public SmileyView(Context context) {  
        super(context, null, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        super(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
    }  
  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)  
    {  
        // Initialize your points: the false objects are the circles of your virtual  
        // objects are rendered (e.g. color, fill and stroke style, etc). Here we are  
        // for the circle and one black stroke point for the eyes and the mouth.  
        // and then in the Activity:  
        // and finally from the onActivityResult get the payment response  
    }  
  
    private void initPaints() {  
        mSmileyPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        mSmileyPaint.setAntiAlias(true);  
        mSmileyPaint.setColor(Color.YELLOW);  
        mSmileyPaint.setStrokeWidth(2);  
        mSmileyPaint.setStyle(Paint.Style.STROKE);  
        mSmileyPaint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));  
        mSmileyPaint.setAlpha(100);  
        mSmileyPaint.setDither(true);  
    }  
  
    3. Implement your own onMeasure(...) method: this is required so that the parent layouts (A8  
        and A9) can measure the view correctly.  
  
    Android Notes for Professionals
```

Chapter 91: Menu

Parameter
`inflate(int resources, Menu menu)`
`getMenuInflater()`
Description
Inflate a menu hierarchy from the specified XML resource.
Returns a `MenuItemLister` with this context.
Initializes the contents of the Activity's standard options menu.
You should place your menu items in to menu.
This method is called whenever an item in your options menu is selected.

Section 91.1: Options menu with dividers

In Android there is a default options menu, which can take a number of options. If a larger number of options needs to be displayed, then it makes sense to group those options in order to maintain clarity. Options can be grouped by putting dividers (i.e. horizontal lines) between them. In order to allow for dividers, the following theme can be used:

```
1. Create your custom view skeleton: this is basically the same for every custom view.  
sketches for a custom view that can draw a smiley, called SmileyView.  
  
public class SmileyView extends View {  
    private Paint mPaint;  
    private Paint mSmileyPaint;  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius = new RectF();  
  
    public SmileyView(Context context) {  
        super(context, null, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        super(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
    }  
  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)  
    {  
        // Initialize your points: the false objects are the circles of your virtual  
        // objects are rendered (e.g. color, fill and stroke style, etc). Here we are  
        // for the circle and one black stroke point for the eyes and the mouth.  
        // and then in the Activity:  
        // and finally from the onActivityResult get the payment response  
    }  
  
    private void initPaints() {  
        mSmileyPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        mSmileyPaint.setAntiAlias(true);  
        mSmileyPaint.setColor(Color.YELLOW);  
        mSmileyPaint.setStrokeWidth(2);  
        mSmileyPaint.setStyle(Paint.Style.STROKE);  
        mSmileyPaint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));  
        mSmileyPaint.setAlpha(100);  
        mSmileyPaint.setDither(true);  
    }  
  
    3. Implement your own onMeasure(...) method: this is required so that the parent layouts (A8  
        and A9) can measure the view correctly.  
  
    Android Notes for Professionals
```

Chapter 159: Android PayPal Gateway Integration

Section 159.1: Setup PayPal in your android code

1) First go through PayPal Developer web site and create an application.
2) Now open your manifest file and give the below permissions:

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3) And some required Activity and Services

```
service  
    android:name=".PaypalService"  
    android:exported="false"/>  
    private Paint mPaypalPaint;  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius = new RectF();  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        super(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
    }  
  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)  
    {  
        // Initialize your points: the false objects are the circles of your virtual  
        // objects are rendered (e.g. color, fill and stroke style, etc). Here we are  
        // for the circle and one black stroke point for the eyes and the mouth.  
        // and then in the Activity:  
        // and finally from the onActivityResult get the payment response  
    }  
  
    private void initPaints() {  
        mPaypalPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        mPaypalPaint.setAntiAlias(true);  
        mPaypalPaint.setColor(Color.BLUE);  
        mPaypalPaint.setStrokeWidth(2);  
        mPaypalPaint.setStyle(Paint.Style.STROKE);  
        mPaypalPaint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));  
        mPaypalPaint.setAlpha(100);  
        mPaypalPaint.setDither(true);  
    }  
  
    3. Implement your own onMeasure(...) method: this is required so that the parent layouts (A8  
        and A9) can measure the view correctly.  
  
    Android Notes for Professionals
```

1000+ 页
专业提示和技巧

1000+ pages
of professional hints and tricks

目录

关于	1
第1章：Android入门	2
第1.1节：创建新项目	2
第1.2节：设置Android Studio	13
第1.3节：无IDE的安卓编程	14
第1.4节：应用基础	18
第1.5节：设置AVD（安卓虚拟设备）	19
第2章：Android Studio	23
第2.1节：设置Android Studio	23
第2.2节：在Android Studio中查看和添加快捷方式	23
第2.3节：Android Studio实用快捷方式	24
第2.4节：提升Android Studio性能的技巧	25
第2.5节：Gradle构建项目耗时过长	26
第2.6节：启用/禁用空行复制	26
第2.7节：基于消息重要性定制logcat消息颜色	27
第2.8节：从界面过滤日志	28
第2.9节：创建过滤器配置	29
第2.10节：创建资源文件夹	30
第3章：Android Studio中的即时运行	32
第3.1节：启用或禁用即时运行	32
第3.2节：即时运行中的代码交换类型	32
第3.3节：使用即时运行时不支持的代码更改	33
第4章：文本视图	34
第4.1节：可变文本视图	34
第4.2节：带删除线的TextView	35
第4.3节：带图片的TextView	36
第4.4节：使RelativeSizeSpan顶部对齐	36
第4.5节：TextView上的捏合缩放	38
第4.6节：具有不同文本大小的文本视图	39
第4.7节：主题和样式自定义	39
第4.8节：TextView自定义	41
第4.9节：单个TextView显示两种不同颜色	44
第5章：自动完成文本框（AutoCompleteTextView）	46
第5.1节：带自定义适配器、点击监听器和过滤器的自动完成	46
第5.2节：简单的硬编码AutoCompleteTextView	49
第6章：自动调整大小的TextView	50
第6.1节：粒度	50
第6.2节：预设尺寸	50
第7章：ListView	52
第7.1节：自定义 ArrayAdapter	52
第7.2节：使用ArrayAdapter的基本ListView	53
第7.3节：使用CursorAdapter进行过滤	53
第8章：布局	55
第8.1节：布局参数	55
第8.2节：重力和布局重力	58
第8.3节：CoordinatorLayout滚动行为	60

Contents

About	1
Chapter 1: Getting started with Android	2
Section 1.1: Creating a New Project	2
Section 1.2: Setting up Android Studio	13
Section 1.3: Android programming without an IDE	14
Section 1.4: Application Fundamentals	18
Section 1.5: Setting up an AVD (Android Virtual Device)	19
Chapter 2: Android Studio	23
Section 2.1: Setup Android Studio	23
Section 2.2: View And Add Shortcuts in Android Studio	23
Section 2.3: Android Studio useful shortcuts	24
Section 2.4: Android Studio Improve performance tip	25
Section 2.5: Gradle build project takes forever	26
Section 2.6: Enable/Disable blank line copy	26
Section 2.7: Custom colors of logcat message based on message importance	27
Section 2.8: Filter logs from UI	28
Section 2.9: Create filters configuration	29
Section 2.10: Create assets folder	30
Chapter 3: Instant Run in Android Studio	32
Section 3.1: Enabling or disabling Instant Run	32
Section 3.2: Types of code Swaps in Instant Run	32
Section 3.3: Unsupported code changes when using Instant Run	33
Chapter 4: TextView	34
Section 4.1: Spannable TextView	34
Section 4.2: Strikethrough TextView	35
Section 4.3: TextView with image	36
Section 4.4: Make RelativeSizeSpan align to top	36
Section 4.5: Pinchzoom on TextView	38
Section 4.6: Textview with different Textsize	39
Section 4.7: Theme and Style customization	39
Section 4.8: TextView customization	41
Section 4.9: Single TextView with two different colors	44
Chapter 5: AutoCompleteTextView	46
Section 5.1: AutoComplete with CustomAdapter, ClickListener and Filter	46
Section 5.2: Simple, hard-coded AutoCompleteTextView	49
Chapter 6: Autosizing TextViews	50
Section 6.1: Granularity	50
Section 6.2: Preset Sizes	50
Chapter 7: ListView	52
Section 7.1: Custom ArrayAdapter	52
Section 7.2: A basic ListView with an ArrayAdapter	53
Section 7.3: Filtering with CursorAdapter	53
Chapter 8: Layouts	55
Section 8.1: LayoutParams	55
Section 8.2: Gravity and layout gravity	58
Section 8.3: CoordinatorLayout Scrolling Behavior	60

第8.4节：百分比布局	62
第8.5节：视图权重	63
第8.6节：编程创建LinearLayout	64
第8.7节：LinearLayout	65
第8.8节：RelativeLayout	66
第8.9节：FrameLayout	68
第8.10节：GridLayout	69
第8.11节：CoordinatorLayout	71
第9章：ConstraintLayout	73
第9.1节：将ConstraintLayout添加到您的项目中	73
第9.2节：链	74
第10章：TextInputLayout	75
第10.1节：基本用法	75
第10.2节：密码可见性切换	75
第10.3节：添加字符计数	75
第10.4节：错误处理	76
第10.5节：自定义TextInputLayout的外观	76
第10.6节：EditText	77
第11章：CoordinatorLayout与行为	79
第11.1节：创建一个简单的行为	79
第11.2节：使用SwipeDismissBehavior	80
第11.3节：创建视图之间的依赖关系	80
第12章：TabLayout	82
第12.1节：在没有ViewPager的情况下使用TabLayout	82
第13章：ViewPager	83
第13.1节：带点指示器的ViewPager	83
第13.2节：带有碎片的基本ViewPager用法	85
第13.3节：带有PreferenceFragment的ViewPager	86
第13.4节：添加ViewPager	87
第13.5节：设置OnPageChangeListener	88
第13.6节：带有TabLayout的ViewPager	89
第14章：CardView	92
第14.1节：CardView入门	92
第14.2节：添加涟漪动画	93
第14.3节：自定义CardView	93
第14.4节：在CardView中使用图片作为背景（Lollipop之前设备的问题）	94
第14.5节：使用TransitionDrawable动画CardView背景颜色	96
第15章：导航视图	97
第15.1节：如何添加NavigationView	97
第15.2节：在菜单元素中添加下划线	101
第15.3节：向菜单添加分隔符	102
第15.4节：使用默认的DividerItemDecoration添加菜单分隔线	103
第16章：RecyclerView	105
第16.1节：添加RecyclerView	105
第16.2节：更流畅的项目加载	106
第16.3节：带DataBinding的RecyclerView	107
第16.4节：动画数据变化	108
第16.5节：带有RecyclerView的弹出菜单	112
第16.6节：使用带有ItemViewType的多个ViewHolder	114

Section 8.4: Percent Layouts	62
Section 8.5: View Weight	63
Section 8.6: Creating LinearLayout programmatically	64
Section 8.7: LinearLayout	65
Section 8.8: RelativeLayout	66
Section 8.9: FrameLayout	68
Section 8.10: GridLayout	69
Section 8.11: CoordinatorLayout	71
Chapter 9: ConstraintLayout	73
Section 9.1: Adding ConstraintLayout to your project	73
Section 9.2: Chains	74
Chapter 10: TextInputLayout	75
Section 10.1: Basic usage	75
Section 10.2: Password Visibility Toggles	75
Section 10.3: Adding Character Counting	75
Section 10.4: Handling Errors	76
Section 10.5: Customizing the appearance of the TextInputLayout	76
Section 10.6: TextInputEditText	77
Chapter 11: CoordinatorLayout and Behaviors	79
Section 11.1: Creating a simple Behavior	79
Section 11.2: Using the SwipeDismissBehavior	80
Section 11.3: Create dependencies between Views	80
Chapter 12: TabLayout	82
Section 12.1: Using a TabLayout without a ViewPager	82
Chapter 13: ViewPager	83
Section 13.1: ViewPager with a dots indicator	83
Section 13.2: Basic ViewPager usage with fragments	85
Section 13.3: ViewPager with PreferenceFragment	86
Section 13.4: Adding a ViewPager	87
Section 13.5: Setup OnPageChangeListener	88
Section 13.6: ViewPager with TabLayout	89
Chapter 14: CardView	92
Section 14.1: Getting Started with CardView	92
Section 14.2: Adding Ripple animation	93
Section 14.3: Customizing the CardView	93
Section 14.4: Using Images as Background in CardView (Pre-Lollipop device issues)	94
Section 14.5: Animate CardView background color with TransitionDrawable	96
Chapter 15: NavigationView	97
Section 15.1: How to add the NavigationView	97
Section 15.2: Add underline in menu elements	101
Section 15.3: Add separators to menu	102
Section 15.4: Add menu Divider using default DividerItemDecoration	103
Chapter 16: RecyclerView	105
Section 16.1: Adding a RecyclerView	105
Section 16.2: Smoother loading of items	106
Section 16.3: RecyclerView with DataBinding	107
Section 16.4: Animate data change	108
Section 16.5: Popup menu with recyclerView	112
Section 16.6: Using several ViewHolders with ItemViewType	114

第16.7节：使用SearchView过滤RecyclerView中的项目	115	Section 16.7: Filter items inside RecyclerView with a SearchView	115
第16.8节：RecyclerView中的拖拽和滑动	116	Section 16.8: Drag&Drop and Swipe with RecyclerView	116
第16.9节：在项目加载前或数据不可用时显示默认视图	117	Section 16.9: Show default view till items load or when data is not available	117
第16.10节：为RecyclerView添加头部/尾部	119	Section 16.10: Add header/footer to a RecyclerView	119
第16.11节：RecyclerView中的无限滚动	122	Section 16.11: Endless Scrolling in Recycleview	122
第16.12节：为RecyclerView项目添加分割线	122	Section 16.12: Add divider lines to RecyclerView items	122
第17章：RecyclerView 装饰	125	Chapter 17: RecyclerView Decorations	125
第17.1节：向 RecyclerView 添加分隔线	125	Section 17.1: Add divider to RecyclerView	125
第17.2节：绘制分隔符	127	Section 17.2: Drawing a Separator	127
第17.3节：如何使用 DividerItemDecoration 添加分隔线	128	Section 17.3: How to add dividers using and DividerItemDecoration	128
第17.4节：使用 ItemDecoration 设置每个项目的边距	128	Section 17.4: Per-item margins with ItemDecoration	128
第17.5节：RecycleView中GridLayoutManager的ItemOffsetDecoration	129	Section 17.5: ItemOffsetDecoration for GridLayoutManager in RecycleView	129
第18章：RecyclerView 点击监听器	131	Chapter 18: RecyclerView onClickListeners	131
第18.1节：Kotlin 和 RxJava 示例	131	Section 18.1: Kotlin and RxJava example	131
第18.2节：RecyclerView 点击监听器	132	Section 18.2: RecyclerView Click listener	132
第18.3节：实现项目点击监听器的另一种方法	133	Section 18.3: Another way to implement Item Click Listener	133
第18.4节：新示例	135	Section 18.4: New Example	135
第18.5节：简单的OnLongClick和OnClick示例	136	Section 18.5: Easy OnLongClick and OnClick Example	136
第18.6节：项目点击监听器	139	Section 18.6: Item Click Listeners	139
第19章：RecyclerView和布局管理器	141	Chapter 19: RecyclerView and LayoutManagers	141
第19.1节：向使用GridLayoutManager的RecyclerView添加头部视图	141	Section 19.1: Adding header view to recyclerview with gridlayout manager	141
第19.2节：具有动态跨度计数的GridLayoutManager	142	Section 19.2: GridLayoutManager with dynamic span count	142
第19.3节：使用LinearLayoutManager的简单列表	144	Section 19.3: Simple list with LinearLayoutManager	144
第19.4节：错列网格布局管理器 (StaggeredGridLayoutManager)	148	Section 19.4: StaggeredGridLayoutManager	148
第20章：RecyclerView中的分页	151	Chapter 20: Pagination in RecyclerView	151
第20.1节：MainActivity.java	151	Section 20.1: MainActivity.java	151
第21章：ImageView	156	Chapter 21: ImageView	156
第21.1节：设置色调	156	Section 21.1: Set tint	156
第21.2节：设置透明度	157	Section 21.2: Set alpha	157
第21.3节：设置缩放类型	157	Section 21.3: Set Scale Type	157
第21.4节：ImageView 缩放类型 - 居中	162	Section 21.4: ImageView ScaleType - Center	162
第21.5节：ImageView 缩放类型 - CenterCrop	164	Section 21.5: ImageView ScaleType - CenterCrop	164
第21.6节：ImageView 缩放类型 - CenterInside	166	Section 21.6: ImageView ScaleType - CenterInside	166
第21.7节：ImageView 缩放类型 - FitStart 和 FitEnd	168	Section 21.7: ImageView ScaleType - FitStart and FitEnd	168
第21.8节：ImageView 缩放类型 - FitCenter	172	Section 21.8: ImageView ScaleType - FitCenter	172
第21.9节：设置图像资源	174	Section 21.9: Set Image Resource	174
第21.10节：ImageView 缩放类型 - FitXY	175	Section 21.10: ImageView ScaleType - FitXY	175
第21.11节：MLRoundedImageView.java	177	Section 21.11: MLRoundedImageView.java	177
第22章：VideoView	180	Chapter 22: VideoView	180
第22.1节：使用 VideoView 从 URL 播放视频	180	Section 22.1: Play video from URL with using VideoView	180
第22.2节：VideoView 创建	180	Section 22.2: VideoView Create	180
第23章：优化的视频视图	181	Chapter 23: Optimized VideoView	181
第23.1节：ListView中的优化视频视图	181	Section 23.1: Optimized VideoView in ListView	181
第24章：网页视图	193	Chapter 24: WebView	193
第24.1节：通过打印控制台消息或远程调试排查网页视图问题	193	Section 24.1: Troubleshooting WebView by printing console messages or by remote debugging	193
第24.2节：从Javascript到Java (Android) 的通信	194	Section 24.2: Communication from Javascript to Java (Android)	194
第24.3节：从Java到JavaScript的通信	195	Section 24.3: Communication from Java to Javascript	195
第24.4节：打开拨号器示例	195	Section 24.4: Open dialer example	195
第24.5节：打开本地文件 / 在WebView中创建动态内容	196	Section 24.5: Open Local File / Create dynamic content in Webview	196

第24.6节：WebView中的JavaScript警告对话框 - 如何使其正常工作	196
第25章：搜索视图（SearchView）	198
第25.1节：为搜索视图设置主题	198
第25.2节：带有Fragment的工具栏中的SearchView	198
第25.3节：使用RxBindings观察器的Appcompat SearchView	200
第26章：底部导航视图（BottomNavigationView）	203
第26.1节：基本实现	203
第26.2节：底部导航视图的自定义	204
第26.3节：处理启用/禁用状态	204
第26.4节：允许超过3个菜单	205
第27章：使用SurfaceView进行画布绘制	207
第27.1节：带绘图线程的SurfaceView	207
第28章：创建自定义视图	212
第28.1节：创建自定义视图	212
第28.2节：向视图添加属性	214
第28.3节：自定义视图性能技巧	216
第28.4节：创建复合视图	217
第28.5节：将SVG/VectorDrawable作为drawableRight的复合视图	220
第28.6节：响应触摸事件	223
第29章：获取计算后的视图尺寸	224
第29.1节：在Activity中计算初始视图尺寸	224
第30章：向Android项目添加FuseView	225
第30.1节：hikr应用程序，只是另一个android.view.View	225
第31章：支持不同分辨率、尺寸的屏幕	232
第31.1节：使用配置限定符	232
第31.2节：将dp和sp转换为像素	232
第31.3节：文本大小和不同的Android屏幕尺寸	233
第32章：ViewFlipper	234
第32.1节：带图片滑动的ViewFlipper	234
第33章：设计模式	235
第33.1节：观察者模式	235
第33.2节：单例类示例	235
第34章：Activity	237
第34.1节：Activity的launchMode	237
第34.2节：将Activity排除在返回栈历史之外	238
第34.3节：Android Activity生命周期详解	238
第34.4节：结束应用并从最近任务中排除	241
第34.5节：使用setContentView呈现界面	242
第34.6节：活动的向上导航	243
第34.7节：清除当前的活动栈并启动一个新活动	244
第35章：活动识别	246
第35.1节：Google Play ActivityRecognitionAPI	246
第35.2节：PathSense活动识别	248
第36章：分屏/多屏活动	250
第36.1节：在Android Nougat中引入的分屏功能实现	250
第37章：材质设计	251
第37.1节：添加工具栏	251
第37.2节：使用材质设计样式的按钮	252

Section 24.6: JavaScript alert dialogs in WebView - How to make them work	196
Chapter 25: SearchView	198
Section 25.1: Setting Theme for SearchView	198
Section 25.2: SearchView in Toolbar with Fragment	198
Section 25.3: Appcompat SearchView with RxBindings watcher	200
Chapter 26: BottomNavigationView	203
Section 26.1: Basic implemetation	203
Section 26.2: Customization of BottomNavigationView	204
Section 26.3: Handling Enabled / Disabled states	204
Section 26.4: Allowing more than 3 menus	205
Chapter 27: Canvas drawing using SurfaceView	207
Section 27.1: SurfaceView with drawing thread	207
Chapter 28: Creating Custom Views	212
Section 28.1: Creating Custom Views	212
Section 28.2: Adding attributes to views	214
Section 28.3: CustomView performance tips	216
Section 28.4: Creating a compound view	217
Section 28.5: Compound view for SVG/VectorDrawable as drawableRight	220
Section 28.6: Responding to Touch Events	223
Chapter 29: Getting Calculated View Dimensions	224
Section 29.1: Calculating initial View dimensions in an Activity	224
Chapter 30: Adding a FuseView to an Android Project	225
Section 30.1: hikr app, just another android.view.View	225
Chapter 31: Supporting Screens With Different Resolutions, Sizes	232
Section 31.1: Using configuration qualifiers	232
Section 31.2: Converting dp and sp to pixels	232
Section 31.3: Text size and different android screen sizes	233
Chapter 32: ViewFlipper	234
Section 32.1: ViewFlipper with image sliding	234
Chapter 33: Design Patterns	235
Section 33.1: Observer pattern	235
Section 33.2: Singleton Class Example	235
Chapter 34: Activity	237
Section 34.1: Activity launchMode	237
Section 34.2: Exclude an activity from back-stack history	238
Section 34.3: Android Activity LifeCycle Explained	238
Section 34.4: End Application with exclude from Recents	241
Section 34.5: Presenting UI with setContentView	242
Section 34.6: Up Navigation for Activities	243
Section 34.7: Clear your current Activity stack and launch a new Activity	244
Chapter 35: Activity Recognition	246
Section 35.1: Google Play ActivityRecognitionAPI	246
Section 35.2: PathSense Activity Recognition	248
Chapter 36: Split Screen / Multi-Screen Activities	250
Section 36.1: Split Screen introduced in Android Nougat implemented	250
Chapter 37: Material Design	251
Section 37.1: Adding a Toolbar	251
Section 37.2: Buttons styled with Material Design	252

第37.3节：添加悬浮操作按钮 (FAB)	253
第37.4节：RippleDrawable	254
第37.5节：添加TabLayout	259
第37.6节：设计支持库中的底部面板	261
第37.7节：应用AppCompat主题	264
第37.8节：添加Snackbar	265
第37.9节：添加导航抽屉	266
第37.10节：如何使用TextInputLayout	269
第38章：资源	270
第38.1节：定义颜色	270
第38.2节：颜色透明度 (Alpha) 级别	271
第38.3节：定义字符串复数形式	271
第38.4节：定义字符串	272
第38.5节：定义尺寸	273
第38.6节：strings.xml中的字符串格式化	273
第38.7节：定义整型数组	274
第38.8节：定义颜色状态列表	274
第38.9节：9补丁	275
第38.10节：获取资源而无“已弃用”警告	278
第38.11节：使用strings.xml文件	278
第38.12节：定义字符串数组	279
第38.13节：定义整数	280
第38.14节：定义菜单资源并在Activity/Fragment中使用	280
第39章：数据绑定库	282
第39.1节：基本文本字段绑定	282
第39.2节：内置的双向数据绑定	283
第39.3节：使用Lambda表达式的自定义事件	284
第39.4节：数据绑定中的默认值	286
第39.5节：对话框中的数据绑定	286
第39.6节：使用访问器方法的绑定	286
第39.7节：在BindingAdapter中以引用方式传递控件	287
第39.8节：带绑定的点击监听器	288
第39.9节：RecyclerView适配器中的数据绑定	289
第39.10节：Fragment中的数据绑定	290
第39.11节：带自定义变量 (int, boolean) 的数据绑定	291
第39.12节：引用类	291
第40章：SharedPreferences	293
第40.1节：使用SharedPreferences实现设置界面	293
第40.2节：Commit与Apply	295
第40.3节：读写SharedPreferences中的值	295
第40.4节：从特定SharedPreferences文件中检索所有存储的条目	296
第40.5节：使用单例读取和写入SharedPreferences数据	297
第40.6节：getPreferences(int) 与 getSharedPreferences(String, int) 的比较	301
第40.7节：监听SharedPreferences的变化	301
第40.8节：在SharedPreferences中存储、检索、移除和清除数据	302
第40.9节：为EditTextPreference添加过滤器	302
第40.10节：SharedPreferences中支持的数据类型	303
第40.11节：实例化SharedPreferences对象的不同方式	303
第40.12节：移除键	304
第40.13节：支持Honeycomb之前版本的StringSet	304

Section 37.3: Adding a FloatingActionButton (FAB)	253
Section 37.4: RippleDrawable	254
Section 37.5: Adding a TabLayout	259
Section 37.6: Bottom Sheets in Design Support Library	261
Section 37.7: Apply an AppCompat theme	264
Section 37.8: Add a Snackbar	265
Section 37.9: Add a Navigation Drawer	266
Section 37.10: How to use TextInputLayout	269
Chapter 38: Resources	270
Section 38.1: Define colors	270
Section 38.2: Color Transparency(Alpha) Level	271
Section 38.3: Define String Plurals	271
Section 38.4: Define strings	272
Section 38.5: Define dimensions	273
Section 38.6: String formatting in strings.xml	273
Section 38.7: Define integer array	274
Section 38.8: Define a color state list	274
Section 38.9: 9 Patches	275
Section 38.10: Getting resources without "deprecated" warnings	278
Section 38.11: Working with strings.xml file	278
Section 38.12: Define string array	279
Section 38.13: Define integers	280
Section 38.14: Define a menu resource and use it inside Activity/Fragment	280
Chapter 39: Data Binding Library	282
Section 39.1: Basic text field binding	282
Section 39.2: Built-in two-way Data Binding	283
Section 39.3: Custom event using lambda expression	284
Section 39.4: Default value in Data Binding	286
Section 39.5: Databinding in Dialog	286
Section 39.6: Binding with an accessor method	286
Section 39.7: Pass widget as reference in BindingAdapter	287
Section 39.8: Click listener with Binding	288
Section 39.9: Data binding in RecyclerView Adapter	289
Section 39.10: Databinding in Fragment	290
Section 39.11: DataBinding with custom variables(int,boolean)	291
Section 39.12: Referencing classes	291
Chapter 40: SharedPreferences	293
Section 40.1: Implementing a Settings screen using SharedPreferences	293
Section 40.2: Commit vs. Apply	295
Section 40.3: Read and write values to SharedPreferences	295
Section 40.4: Retrieve all stored entries from a particular SharedPreferences file	296
Section 40.5: Reading and writing data to SharedPreferences with Singleton	297
Section 40.6: getPreferences(int) VS getSharedPreferences(String, int)	301
Section 40.7: Listening for SharedPreferences changes	301
Section 40.8: Store, Retrieve, Remove and Clear Data from SharedPreferences	302
Section 40.9: Add filter for EditTextPreference	302
Section 40.10: Supported data types in SharedPreferences	303
Section 40.11: Different ways of instantiating an object of SharedPreferences	303
Section 40.12: Removing keys	304
Section 40.13: Support pre-Honeycomb with StringSet	304

第41章：Intent（意图）	306
第41.1节：从另一个Activity获取结果	306
第41.2节：在活动之间传递数据	308
第41.3节：在浏览器中打开URL	309
第41.4节：启动器模式	310
第41.5节：清除活动栈	311
第41.6节：启动活动	311
第41.7节：发送电子邮件	312
第41.8节：用于Chrome自定义标签的CustomTabsIntent	312
第41.9节：Intent URI	313
第41.10节：启动拨号器	314
第41.11节：向其他组件广播消息	314
第41.12节：在活动之间传递自定义对象	315
第41.13节：使用指定的纬度和经度打开谷歌地图	317
第41.14节：通过Intent在Activity中传递不同的数据	317
第41.15节：共享Intent	319
第41.16节：显示文件选择器并读取结果	319
第41.17节：通过Intent共享多个文件	321
第41.18节：使用Intent启动无绑定服务	321
第41.19节：从Activity获取结果到Fragment	322
第42章：碎片	324
第42.1节：使用Bundle从Activity传递数据到Fragment	324
第42.2节：newInstance()模式	324
第42.3节：使用回退栈和静态工厂模式在碎片之间导航	325
第42.4节：通过回调接口向Activity发送事件	326
第42.5节：为碎片之间的过渡添加动画	327
第42.6节：碎片之间的通信	328
第43章：按钮	333
第43.1节：在XML中为一个或多个视图使用相同的点击事件	333
第43.2节：定义外部监听器	333
第43.3节：内联onClickListener	334
第43.4节：自定义按钮样式	334
第43.5节：自定义点击监听器以防止多次快速点击	338
第43.6节：使用布局定义点击操作	338
第43.7节：监听长按事件	339
第44章：模拟器	340
第44.1节：截图	340
第44.2节：模拟调用	345
第44.3节：打开AVD管理器	345
第44.4节：启动模拟器时解决错误	345
第45章：服务	347
第45.1节：服务的生命周期	347
第45.2节：定义服务的进程	348
第45.3节：创建无绑定服务	348
第45.4节：启动服务	351
第45.5节：借助Binder创建绑定服务	351
第45.6节：创建远程服务（通过AIDL）	352
第46章：清单文件	354
第46.1节：声明组件	354
第46.2节：在清单文件中声明权限	354

Chapter 41: Intent	306
Section 41.1: Getting a result from another Activity	306
Section 41.2: Passing data between activities	308
Section 41.3: Open a URL in a browser	309
Section 41.4: Starter Pattern	310
Section 41.5: Clearing an activity stack	311
Section 41.6: Start an activity	311
Section 41.7: Sending emails	312
Section 41.8: CustomTabsIntent for Chrome Custom Tabs	312
Section 41.9: Intent URI	313
Section 41.10: Start the dialer	314
Section 41.11: Broadcasting Messages to Other Components	314
Section 41.12: Passing custom object between activities	315
Section 41.13: Open Google map with specified latitude, longitude	317
Section 41.14: Passing different data through Intent in Activity	317
Section 41.15: Share intent	319
Section 41.16: Showing a File Chooser and Reading the Result	319
Section 41.17: Sharing Multiple Files through Intent	321
Section 41.18: Start Unbound Service using an Intent	321
Section 41.19: Getting a result from Activity to Fragment	322
Chapter 42: Fragments	324
Section 42.1: Pass data from Activity to Fragment using Bundle	324
Section 42.2: The newInstance() pattern	324
Section 42.3: Navigation between fragments using backstack and static fabric pattern	325
Section 42.4: Sending events back to an activity with callback interface	326
Section 42.5: Animate the transition between fragments	327
Section 42.6: Communication between Fragments	328
Chapter 43: Button	333
Section 43.1: Using the same click event for one or more Views in the XML	333
Section 43.2: Defining external Listener	333
Section 43.3: inline onClickListener	334
Section 43.4: Customizing Button style	334
Section 43.5: Custom Click Listener to prevent multiple fast clicks	338
Section 43.6: Using the layout to define a click action	338
Section 43.7: Listening to the long click events	339
Chapter 44: Emulator	340
Section 44.1: Taking screenshots	340
Section 44.2: Simulate call	345
Section 44.3: Open the AVD Manager	345
Section 44.4: Resolving Errors while starting emulator	345
Chapter 45: Service	347
Section 45.1: Lifecycle of a Service	347
Section 45.2: Defining the process of a service	348
Section 45.3: Creating an unbound service	348
Section 45.4: Starting a Service	351
Section 45.5: Creating Bound Service with help of Binder	351
Section 45.6: Creating Remote Service (via AIDL)	352
Chapter 46: The Manifest File	354
Section 46.1: Declaring Components	354
Section 46.2: Declaring permissions in your manifest file	354

第47章 : Android的Gradle	356	Chapter 47: Gradle for Android	356
第47.1节 : 一个基本的build.gradle文件	356	Section 47.1: A basic build.gradle file	356
第47.2节 : 定义和使用构建配置字段	358	Section 47.2: Define and use Build Configuration Fields	358
第47.3节 : 通过“dependencies.gradle”文件集中管理依赖项	361	Section 47.3: Centralizing dependencies via "dependencies.gradle" file	361
第47.4节 : 签署APK而不暴露密钥库密码	362	Section 47.4: Sign APK without exposing keystore password	362
第47.5节 : 添加特定产品风味的依赖项	364	Section 47.5: Adding product flavor-specific dependencies	364
第47.6节 : 为构建类型和产品风味指定不同的应用ID	364	Section 47.6: Specifying different application IDs for build types and product flavors	364
第47.7节 : 通过 "version.properties" 文件对构建版本进行版本控制	365	Section 47.7: Versioning your builds via "version.properties" file	365
第47.8节 : 定义产品风味 (product flavors)	366	Section 47.8: Defining product flavors	366
第47.9节 : 更改输出APK名称并添加版本名称 :	366	Section 47.9: Changing output apk name and add version name:	366
第47.10节 : 添加特定产品风味的资源	367	Section 47.10: Adding product flavor-specific resources	367
第47.11节 : 为什么Android Studio项目中有两个build.gradle文件 ?	367	Section 47.11: Why are there two build.gradle files in an Android Studio project?	367
第47.12节 : 风味特定资源的目录结构	368	Section 47.12: Directory structure for flavor-specific resources	368
第47.13节 : 使用gradle启用Proguard	368	Section 47.13: Enable Proguard using gradle	368
第47.14节 : 忽略构建变体	369	Section 47.14: Ignoring build variant	369
第47.15节 : 为Gradle和AndroidStudio启用实验性NDK插件支持	369	Section 47.15: Enable experimental NDK plugin support for Gradle and AndroidStudio	369
第47.16节 : 显示签名信息	371	Section 47.16: Display signing information	371
第47.17节 : 查看依赖树	372	Section 47.17: Seeing dependency tree	372
第47.18节 : 禁用图像压缩以减小APK文件大小	373	Section 47.18: Disable image compression for a smaller APK file size	373
第47.19节 : 自动删除“未对齐”的apk	373	Section 47.19: Delete "unaligned" apk automatically	373
第47.20节 : 从Gradle执行shell脚本	373	Section 47.20: Executing a shell script from gradle	373
第47.21节 : 显示所有Gradle项目任务	374	Section 47.21: Show all gradle project tasks	374
第47.22节 : 调试你的Gradle错误	375	Section 47.22: Debugging your Gradle errors	375
第47.23节 : 使用gradle.properties进行集中版本号/构建配置管理	376	Section 47.23: Use gradle.properties for central versionnumber/buildconfigurations	376
第47.24节 : 定义构建类型	377	Section 47.24: Defining build types	377
第48章 : Android文件输入输出	378	Chapter 48: FileIO with Android	378
第48.1节 : 获取工作文件夹	378	Section 48.1: Obtaining the working folder	378
第48.2节 : 写入原始字节数组	378	Section 48.2: Writing raw array of bytes	378
第48.3节 : 对象序列化	378	Section 48.3: Serializing the object	378
第48.4节 : 写入外部存储 (SD卡)	379	Section 48.4: Writing to external storage (SD card)	379
第48.5节 : 解决“不可见的MTP文件”问题	379	Section 48.5: Solving "Invisible MTP files" problem	379
第48.6节 : 处理大文件	379	Section 48.6: Working with big files	379
第49章 : FileProvider	381	Chapter 49: FileProvider	381
第49.1节 : 共享文件	381	Section 49.1: Sharing a file	381
第50章 : 在内部和外部存储中存储文件	383	Chapter 50: Storing Files in Internal & External Storage	383
第50.1节 : Android : 内部和外部存储——术语说明	383	Section 50.1: Android: Internal and External Storage - Terminology Clarification	383
第50.2节 : 使用外部存储	387	Section 50.2: Using External Storage	387
第50.3节 : 使用内部存储	388	Section 50.3: Using Internal Storage	388
第50.4节 : 获取设备目录 :	388	Section 50.4: Fetch Device Directory :	388
第50.5节 : 将数据库保存到SD卡 (备份数据库到SD)	390	Section 50.5: Save Database on SD Card (Backup DB on SD)	390
第51章 : Android中的压缩文件	392	Chapter 51: Zip file in android	392
第51.1节 : Android上的压缩文件	392	Section 51.1: Zip file on android	392
第52章 : Android中的解压文件	393	Chapter 52: Unzip File in Android	393
第52.1节 : 解压文件	393	Section 52.1: Unzip file	393
第53章 : 相机与图库	394	Chapter 53: Camera and Gallery	394
第53.1节 : 拍照	394	Section 53.1: Take photo	394
第53.2节 : 从相机拍摄全尺寸照片	396	Section 53.2: Taking full-sized photo from camera	396
第53.3节 : 正确解码从意图获取的URI中旋转的位图	399	Section 53.3: Decode bitmap correctly rotated from the uri fetched with the intent	399
第53.4节 : 设置相机分辨率	401	Section 53.4: Set camera resolution	401

第53.5节：如何启动相机或图库并将相机结果保存到存储	401
第54章：相机2 API	405
第54.1节：在TextureView中预览主摄像头	405
第55章：Android中的指纹API	414
第55.1节：如何使用Android指纹API保存用户密码	414
第55.2节：在Android应用中添加指纹扫描仪	421
第56章：蓝牙和蓝牙低功耗（Bluetooth LE）API	424
第56.1节：权限	424
第56.2节：检查蓝牙是否已启用	424
第56.3节：查找附近的低功耗蓝牙设备	424
第56.4节：使设备可被发现	429
第56.5节：连接蓝牙设备	429
第56.6节：查找附近的蓝牙设备	431
第57章：API-23及以上的运行时权限	432
第57.1节：Android 6.0 多重权限	432
第57.2节：来自同一权限组的多个运行时权限	433
第57.3节：使用PermissionUtil	434
第57.4节：将所有与权限相关的代码包含到一个抽象基类中，并扩展该活动的功能基类以实现更简洁/可重用的代码	435
第57.5节：在广播和URI中强制执行权限	437
第58章：Android 地点API	439
第58.1节：使用地点API获取当前位置	439
第58.2节：地点自动完成集成	440
第58.3节：地点选择器使用示例	441
第58.4节：为地点自动完成设置地点类型过滤器	442
第58.5节：添加多个谷歌自动完成活动	443
第59章：Android NDK	445
第59.1节：如何在ndk中进行日志记录	445
第59.2节：为Android构建本地可执行文件	445
第59.3节：如何清理构建	446
第59.4节：如何使用除Android.mk以外的makefile	446
第60章：DayNight主题（AppCompat v23.2 / API 14+）	447
第60.1节：向应用添加DayNight主题	447
第61章：滑翔	448
第61.1节：加载图片	448
第61.2节：将Glide添加到你的项目中	449
第61.3节：Glide圆形变换（在圆形ImageView中加载图片）	449
第61.4节：默认变换	450
第61.5节：使用自定义Glide目标的Glide圆角图片	451
第61.6节：占位符和错误处理	451
第61.7节：预加载图片	452
第61.8节：处理Glide图片加载失败	452
第61.9节：在圆形ImageView中加载图片，无需自定义变换	453
第62章：对话框	454
第62.1节：使用Appcompat向应用添加Material Design风格的AlertDialog	454
第62.2节：一个基本的警告对话框	454
第62.3节：AlertDialog中的ListView	455
第62.4节：带EditText的自定义警告对话框	456
第62.5节：日期选择对话框（DatePickerDialog）	457

Section 53.5: How to start camera or gallery and save camera result to storage	401
Chapter 54: Camera 2 API	405
Section 54.1: Preview the main camera in a TextureView	405
Chapter 55: Fingerprint API in android	414
Section 55.1: How to use Android Fingerprint API to save user passwords	414
Section 55.2: Adding the Fingerprint Scanner in Android application	421
Chapter 56: Bluetooth and Bluetooth LE API	424
Section 56.1: Permissions	424
Section 56.2: Check if bluetooth is enabled	424
Section 56.3: Find nearby Bluetooth Low Energy devices	424
Section 56.4: Make device discoverable	429
Section 56.5: Connect to Bluetooth device	429
Section 56.6: Find nearby bluetooth devices	431
Chapter 57: Runtime Permissions in API-23 +	432
Section 57.1: Android 6.0 multiple permissions	432
Section 57.2: Multiple Runtime Permissions From Same Permission Groups	433
Section 57.3: Using PermissionUtil	434
Section 57.4: Include all permission-related code to an abstract base class and extend the activity of this base class to achieve cleaner/reusable code	435
Section 57.5: Enforcing Permissions in Broadcasts, URI	437
Chapter 58: Android Places API	439
Section 58.1: Getting Current Places by Using Places API	439
Section 58.2: Place Autocomplete Integration	440
Section 58.3: Place Picker Usage Example	441
Section 58.4: Setting place type filters for PlaceAutocomplete	442
Section 58.5: Adding more than one google auto complete activity	443
Chapter 59: Android NDK	445
Section 59.1: How to log in ndk	445
Section 59.2: Building native executables for Android	445
Section 59.3: How to clean the build	446
Section 59.4: How to use a makefile other than Android.mk	446
Chapter 60: DayNight Theme (AppCompat v23.2 / API 14+)	447
Section 60.1: Adding the DayNight theme to an app	447
Chapter 61: Glide	448
Section 61.1: Loading an image	448
Section 61.2: Add Glide to your project	449
Section 61.3: Glide circle transformation (Load image in a circular ImageView)	449
Section 61.4: Default transformations	450
Section 61.5: Glide rounded corners image with custom Glide target	451
Section 61.6: Placeholder and Error handling	451
Section 61.7: Preloading images	452
Section 61.8: Handling Glide image load failed	452
Section 61.9: Load image in a circular ImageView without custom transformations	453
Chapter 62: Dialog	454
Section 62.1: Adding Material Design AlertDialog to your app using Appcompat	454
Section 62.2: A Basic Alert Dialog	454
Section 62.3: ListView in AlertDialog	455
Section 62.4: Custom Alert Dialog with EditText	456
Section 62.5: DatePickerDialog	457

第62章：日期选择器（DatePicker）	457	Section 62.6: DatePicker	457
第62.7节：警告对话框	458	Section 62.7: Alert Dialog	458
第62.8节：带多行标题的警告对话框	459	Section 62.8: Alert Dialog with Multi-line Title	459
第62.9节：DialogFragment中的日期选择器	461	Section 62.9: Date Picker within DialogFragment	461
第62.10节：无背景无标题的全屏自定义对话框	463	Section 62.10: Fullscreen Custom Dialog with no background and no title	463
第63章：增强警告对话框	465	Chapter 63: Enhancing Alert Dialogs	465
第63.1节：包含可点击链接的警告对话框	465	Section 63.1: Alert dialog containing a clickable link	465
第64章：动画警告对话框	466	Chapter 64: Animated AlertDialog Box	466
第64.1节：放置以下代码以实现动画对话框。	466	Section 64.1: Put Below code for Animated dialog.	466
第65章：GreenDAO	469	Chapter 65: GreenDAO	469
第65.1节：SELECT、INSERT、DELETE、UPDATE查询的辅助方法	469	Section 65.1: Helper methods for SELECT, INSERT, DELETE, UPDATE queries	469
第65.2节：使用GreenDAO 3.X创建具有复合主键的实体	471	Section 65.2: Creating an Entity with GreenDAO 3.X that has a Composite Primary Key	471
第65.3节：GreenDao v3.X入门	472	Section 65.3: Getting started with GreenDao v3.X	472
第66章：工具属性	474	Chapter 66: Tools Attributes	474
第66.1节：设计时布局属性	474	Section 66.1: Designtime Layout Attributes	474
第67章：格式化字符串	475	Chapter 67: Formatting Strings	475
第67.1节：格式化字符串资源	475	Section 67.1: Format a string resource	475
第67.2节：数据类型与字符串的相互格式化	475	Section 67.2: Formatting data types to String and vice versa	475
第67.3节：将时间戳格式化为字符串	475	Section 67.3: Format a timestamp to string	475
第68章：可变样式字符串（SpannableString）	476	Chapter 68: SpannableString	476
第68.1节：为TextView添加样式	476	Section 68.1: Add styles to a TextView	476
第68.2节：多串，多色	478	Section 68.2: Multi string , with multi color	478
第69章：通知	480	Chapter 69: Notifications	480
第69.1节：带跑马灯的提醒通知（适用于旧设备）	480	Section 69.1: Heads Up Notification with Ticker for older devices	480
第69.2节：创建简单通知	484	Section 69.2: Creating a simple Notification	484
第69.3节：设置自定义通知 - 显示完整内容文本	484	Section 69.3: Set custom notification - show full content text	484
第69.4节：动态获取大图标的正确像素大小	485	Section 69.4: Dynamically getting the correct pixel size for the large icon	485
第69.5节：带操作按钮的持续通知	485	Section 69.5: Ongoing notification with Action button	485
第69.6节：设置通知的不同优先级	486	Section 69.6: Setting Different priorities in notification	486
第69.7节：使用`Picasso`库设置自定义通知图标	487	Section 69.7: Set custom notification icon using 'Picasso' library	487
第69.8节：调度通知	488	Section 69.8: Scheduling notifications	488
第70章：AlarmManager（闹钟管理器）	490	Chapter 70: AlarmManager	490
第70.1节：如何取消闹钟	490	Section 70.1: How to Cancel an Alarm	490
第70.2节：在所有Android版本上创建精确闹钟	490	Section 70.2: Creating exact alarms on all Android versions	490
第70.3节：API23及以上版本的休眠模式干扰AlarmManager	491	Section 70.3: API23+ Doze mode interferes with AlarmManager	491
第70.4节：稍后运行一个意图	491	Section 70.4: Run an intent at a later time	491
第71章：处理器	492	Chapter 71: Handler	492
第71.1节：HandlerThread和线程间通信	492	Section 71.1: HandlerThreads and communication between Threads	492
第71.2节：使用Handler创建计时器（类似于javax.swing.Timer）	492	Section 71.2: Use Handler to create a Timer (similar to javax.swing.Timer)	492
第71.3节：使用Handler在延迟一段时间后执行代码	493	Section 71.3: Using a Handler to execute code after a delayed amount of time	493
第71.4节：停止处理器执行	494	Section 71.4: Stop handler from execution	494
第72章：BroadcastReceiver	495	Chapter 72: BroadcastReceiver	495
第72.1节：使用LocalBroadcastManager	495	Section 72.1: Using LocalBroadcastManager	495
第72.2节：BroadcastReceiver基础	495	Section 72.2: BroadcastReceiver Basics	495
第72.3节：Broadcast接收器简介	496	Section 72.3: Introduction to Broadcast receiver	496
第72.4节：使用有序广播	496	Section 72.4: Using ordered broadcasts	496
第72.5节：粘性广播	497	Section 72.5: Sticky Broadcast	497
第72.6节：以编程方式启用和禁用广播接收器	497	Section 72.6: Enabling and disabling a Broadcast Receiver programmatically	497
第72.7节：LocalBroadcastManager示例	498	Section 72.7: Example of a LocalBroadcastManager	498

第72.8节：Android停止状态	499
第72.9节：通过自定义广播接收器通信两个活动	499
第72.10节：处理BOOT_COMPLETED事件的BroadcastReceiver	500
第72.11节：蓝牙广播接收器	501
第73章：用户界面生命周期	502
第73.1节：在内存修剪时保存数据	502
第74章：HttpURLConnection	503
第74.1节：创建HttpURLConnection	503
第74.2节：发送HTTP GET请求	503
第74.3节：读取HTTP GET请求的主体	504
第74.4节：发送带参数的HTTP POST请求	504
第74.5节：一个多用途的HttpURLConnection类，用于处理所有类型的HTTP请求	506
第74.6节：使用HttpURLConnection处理multipart/form-data	508
第74.7节：使用HttpURLConnection上传（POST）文件	511
第75章：回调URL	513
第75.1节：使用Instagram OAuth的回调URL示例	513
第76章：Snackbar	514
第76.1节：创建一个简单的Snackbar	514
第76.2节：自定义Snackbar	514
第76.3节：自定义Snackbar（无需视图）	515
第76.4节：带回调的Snackbar	516
第76.5节：Snackbar与Toast：我应该使用哪一个？	516
第76.6节：自定义Snackbar	517
第77章：控件	518
第77.1节：清单声明 -	518
第77.2节：元数据	518
第77.3节：AppWidgetProvider类	518
第77.4节：使用Android Studio创建/集成基础小部件	519
第77.5节：两个不同布局的小部件声明	520
第78章：Toast	522
第78.1节：创建自定义Toast	522
第78.2节：设置Toast的位置	523
第78.3节：显示Toast消息	523
第78.4节：在软键盘上方显示Toast消息	524
第78.5节：线程安全的Toast显示方式（全应用范围）	524
第78.6节：线程安全的Toast消息显示方式（针对 AsyncTask）	525
第79章：为Toast消息创建单例类	526
第79.1节：创建自己的单例类用于吐司消息	526
第80章：接口	528
第80.1节：自定义监听器	528
第80.2节：基础监听器	529
第81章：动画器	531
第81.1节：TransitionDrawable动画	531
第81.2节：淡入/淡出动画	531
第81.3节：ValueAnimator	532
第81.4节：视图的展开和折叠动画	533
第81.5节：ObjectAnimator	534
第81.6节：ViewPropertyAnimator	534
第81.7节：ImageView的抖动动画	535

Section 72.8: Android stopped state	499
Section 72.9: Communicate two activities through custom Broadcast receiver	499
Section 72.10: BroadcastReceiver to handle BOOT_COMPLETED events	500
Section 72.11: Bluetooth Broadcast receiver	501
Chapter 73: UI Lifecycle	502
Section 73.1: Saving data on memory trimming	502
Chapter 74: HttpURLConnection	503
Section 74.1: Creating an HttpURLConnection	503
Section 74.2: Sending an HTTP GET request	503
Section 74.3: Reading the body of an HTTP GET request	504
Section 74.4: Sending an HTTP POST request with parameters	504
Section 74.5: A multi-purpose HttpURLConnection class to handle all types of HTTP requests	506
Section 74.6: Use HttpURLConnection for multipart/form-data	508
Section 74.7: Upload (POST) file using HttpURLConnection	511
Chapter 75: Callback URL	513
Section 75.1: Callback URL example with Instagram OAuth	513
Chapter 76: Snackbar	514
Section 76.1: Creating a simple Snackbar	514
Section 76.2: Custom Snack Bar	514
Section 76.3: Custom Snackbar (no need view)	515
Section 76.4: Snackbar with Callback	516
Section 76.5: Snackbar vs Toasts: Which one should I use?	516
Section 76.6: Custom Snackbar	517
Chapter 77: Widgets	518
Section 77.1: Manifest Declaration -	518
Section 77.2: Metadata	518
Section 77.3: AppWidgetProvider Class	518
Section 77.4: Create/Integrate Basic Widget using Android Studio	519
Section 77.5: Two widgets with different layouts declaration	520
Chapter 78: Toast	522
Section 78.1: Creating a custom Toast	522
Section 78.2: Set position of a Toast	523
Section 78.3: Showing a Toast Message	523
Section 78.4: Show Toast Message Above Soft Keyboard	524
Section 78.5: Thread safe way of displaying Toast (Application Wide)	524
Section 78.6: Thread safe way of displaying a Toast Message (For AsyncTask)	525
Chapter 79: Create Singleton Class for Toast Message	526
Section 79.1: Create own singleton class for toast massages	526
Chapter 80: Interfaces	528
Section 80.1: Custom Listener	528
Section 80.2: Basic Listener	529
Chapter 81: Animators	531
Section 81.1: TransitionDrawable animation	531
Section 81.2: Fade in/out animation	531
Section 81.3: ValueAnimator	532
Section 81.4: Expand and Collapse animation of View	533
Section 81.5: ObjectAnimator	534
Section 81.6: ViewPropertyAnimator	534
Section 81.7: Shake animation of an ImageView	535

第82章：位置	537	Chapter 82: Location 537
第82.1节：融合位置API	537	Section 82.1: Fused location API 537
第82.2节：使用地理编码器从位置获取地址	541	Section 82.2: Get Address From Location using Geocoder 541
第82.3节：使用LocationManager请求位置更新	542	Section 82.3: Requesting location updates using LocationManager 542
第82.4节：使用LocationManager在单独线程请求位置更新	543	Section 82.4: Requesting location updates on a separate thread using LocationManager 543
第82.5节：在BroadcastReceiver中获取位置更新	544	Section 82.5: Getting location updates in a BroadcastReceiver 544
第82.6节：注册地理围栏	545	Section 82.6: Register geofence 545
第83章：主题、样式、属性	549	Chapter 83: Theme, Style, Attribute 549
第83.1节：定义主色、主色暗色和强调色	549	Section 83.1: Define primary, primary dark, and accent colors 549
第83.2节：一个应用中的多主题	549	Section 83.2: Multiple Themes in one App 549
第83.3节：导航栏颜色 (API 21及以上)	551	Section 83.3: Navigation Bar Color (API 21+) 551
第83.4节：为每个活动使用自定义主题	551	Section 83.4: Use Custom Theme Per Activity 551
第83.5节：浅色状态栏 (API 23及以上)	552	Section 83.5: Light Status Bar (API 23+) 552
第83.6节：全局使用自定义主题	552	Section 83.6: Use Custom Theme Globally 552
第83.7节：过度滚动颜色 (API 21及以上)	552	Section 83.7: Overscroll Color (API 21+) 552
第83.8节：涟漪颜色 (API 21及以上)	552	Section 83.8: Ripple Color (API 21+) 552
第83.9节：半透明导航栏和状态栏 (API 19及以上)	553	Section 83.9: Translucent Navigation and Status Bars (API 19+) 553
第83.10节：主题继承	553	Section 83.10: Theme inheritance 553
第84章：MediaPlayer	554	Chapter 84: MediaPlayer 554
第84.1节：基本创建与播放	554	Section 84.1: Basic creation and playing 554
第84.2节：带缓冲进度和播放位置的媒体播放器	554	Section 84.2: Media Player with Buffer progress and play position 554
第84.3节：获取系统铃声	556	Section 84.3: Getting system ringtones 556
第84.4节：异步准备	557	Section 84.4: Asynchronous prepare 557
第84.5节：导入音频到Android Studio并播放	557	Section 84.5: Import audio into androidstudio and play it 557
第84.6节：获取和设置系统音量	559	Section 84.6: Getting and setting system volume 559
第85章：Android声音与媒体	561	Chapter 85: Android Sound and Media 561
第85.1节：如何为API大于19选择图片和视频	561	Section 85.1: How to pick image and video for api >19 561
第85.2节：通过SoundPool播放声音	562	Section 85.2: Play sounds via SoundPool 562
第86章：MediaSession	563	Chapter 86: MediaSession 563
第86.1节：接收和处理按钮事件	563	Section 86.1: Receiving and handling button events 563
第87章：MediaStore	566	Chapter 87: MediaStore 566
第87.1节：从设备的特定文件夹获取音频/MP3文件或获取所有文件	566	Section 87.1: Fetch Audio/MP3 files from specific folder of device or fetch all files 566
第88章：Multidex与Dex方法限制	569	Chapter 88: Multidex and the Dex Method Limit 569
第88.1节：启用Multidex	569	Section 88.1: Enabling Multidex 569
第88.2节：通过扩展Application实现Multidex	570	Section 88.2: Multidex by extending Application 570
第88.3节：通过扩展MultiDexApplication实现Multidex	570	Section 88.3: Multidex by extending MultiDexApplication 570
第88.4节：直接使用MultiDexApplication实现Multidex	571	Section 88.4: Multidex by using MultiDexApplication directly 571
第88.5节：每次构建时统计方法引用数 (Dexcount Gradle插件)	571	Section 88.5: Counting Method References On Every Build (Dexcount Gradle Plugin) 571
第89章：使用Sync Adapter进行数据同步	573	Chapter 89: Data Synchronization with Sync Adapter 573
第89.1节：带有Stub Provider的虚拟Sync Adapter	573	Section 89.1: Dummy Sync Adapter with Stub Provider 573
第90章：波特杜模式	579	Chapter 90: PorterDuff Mode 579
第90.1节：创建PorterDu颜色滤镜	579	Section 90.1: Creating a PorterDuff ColorFilter 579
第90.2节：创建PorterDu XferMode	579	Section 90.2: Creating a PorterDuff XferMode 579
第90.3节：使用PorterDuXfermode对位图应用径向遮罩 (晕影)	579	Section 90.3: Apply a radial mask (vignette) to a bitmap using PorterDuffXfermode 579
第91章：菜单	581	Chapter 91: Menu 581
第91.1节：带分隔符的选项菜单	581	Section 91.1: Options menu with dividers 581
第91.2节：为菜单应用自定义字体	581	Section 91.2: Apply custom font to Menu 581
第91.3节：在活动中创建菜单	582	Section 91.3: Creating a Menu in an Activity 582
第92章：毕加索	585	Chapter 92: Picasso 585

第92.1节：将毕加索库添加到您的安卓项目中	585
第92.2节：使用毕加索实现圆形头像	585
第92.3节：占位符和错误处理	587
第92.4节：调整大小和旋转	587
第92.5节：禁用毕加索缓存	588
第92.6节：将毕加索用作Html.fromHtml的ImageGetter	588
第92.7节：使用Picasso取消图像请求	589
第92.8节：从外部存储加载图片	590
第92.9节：使用Picasso下载图片为Bitmap	590
第92.10节：先尝试离线磁盘缓存，然后在线获取图片	590
第93章：RoboGuice	592
第93.1节：简单示例	592
第93.2节：Gradle项目的安装	592
第93.3节：@ContentView注解	592
第93.4节：@InjectResource 注解	592
第93.5节：@InjectView 注解	593
第93.6节：RoboGuice 简介	593
第94章：ACRA	596
第94.1节：ACRAHandler	596
第94.2节：示例清单	596
第94.3节：安装	597
第95章：可打包的	598
第95.1节：使自定义对象实现Parcelable接口	598
第95.2节：包含另一个Parcelable对象的Parcelable对象	599
第95.3节：在Parcelable中使用枚举	600
第96章：Retrofit2	602
第96.1节：一个简单的GET请求	602
第96.2节：使用Stetho调试	604
第96.3节：为Retrofit2添加日志记录	605
第96.4节：使用GSON的简单POST请求	605
第96.5节：使用Retrofit2从服务器下载文件	607
第96.6节：使用Retrofit作为多部分上传多个文件	609
第96.7节：带OkHttp拦截器的Retrofit	612
第96.8节：头部和主体：一个认证示例	612
第96.9节：通过Multipart上传文件	613
第96.10节：Retrofit 2自定义Xml转换器	613
第96.11节：使用Retrofit 2读取URL中的XML	615
第97章：ButterKnife	618
第97.1节：在项目中配置ButterKnife	618
第97.2节：ButterKnife中解绑视图	620
第97.3节：使用ButterKnife绑定监听器	620
第97.4节：Android Studio ButterKnife插件	621
第97.5节：使用ButterKnife绑定视图	622
第98章：Volley	625
第98.1节：使用Volley进行HTTP请求	625
第98.2节：使用GET方法的基本StringRequest	626
第98.3节：向NetworkImageView添加自定义设计时属性	627
第98.4节：向请求添加自定义头部（例如用于基本认证）	628
第98.5节：通过POST方法使用StringRequest进行远程服务器认证	629
第98.6节：取消请求	631

Section 92.1: Adding Picasso Library to your Android Project	585
Section 92.2: Circular Avatars with Picasso	585
Section 92.3: Placeholder and Error Handling	587
Section 92.4: Re-sizing and Rotating	587
Section 92.5: Disable cache in Picasso	588
Section 92.6: Using Picasso as ImageGetter for Html.fromHtml	588
Section 92.7: Cancelling Image Requests using Picasso	589
Section 92.8: Loading Image from external Storage	590
Section 92.9: Downloading image as Bitmap using Picasso	590
Section 92.10: Try offline disk cache first, then go online and fetch the image	590
Chapter 93: RoboGuice	592
Section 93.1: Simple example	592
Section 93.2: Installation for Gradle Projects	592
Section 93.3: @ContentView annotation	592
Section 93.4: @InjectResource annotation	592
Section 93.5: @InjectView annotation	593
Section 93.6: Introduction to RoboGuice	593
Chapter 94: ACRA	596
Section 94.1: ACRAHandler	596
Section 94.2: Example manifest	596
Section 94.3: Installation	597
Chapter 95: Parcelable	598
Section 95.1: Making a custom object Parcelable	598
Section 95.2: Parcelable object containing another Parcelable object	599
Section 95.3: Using Enums with Parcelable	600
Chapter 96: Retrofit2	602
Section 96.1: A Simple GET Request	602
Section 96.2: Debugging with Stetho	604
Section 96.3: Add logging to Retrofit2	605
Section 96.4: A simple POST request with GSON	605
Section 96.5: Download a file from Server using Retrofit2	607
Section 96.6: Upload multiple file using Retrofit as multipart	609
Section 96.7: Retrofit with OkHttp interceptor	612
Section 96.8: Header and Body: an Authentication Example	612
Section 96.9: Uploading a file via Multipart	613
Section 96.10: Retrofit 2 Custom Xml Converter	613
Section 96.11: Reading XML form URL with Retrofit 2	615
Chapter 97: ButterKnife	618
Section 97.1: Configuring ButterKnife in your project	618
Section 97.2: Unbinding views in ButterKnife	620
Section 97.3: Binding Listeners using ButterKnife	620
Section 97.4: Android Studio ButterKnife Plugin	621
Section 97.5: Binding Views using ButterKnife	622
Chapter 98: Volley	625
Section 98.1: Using Volley for HTTP requests	625
Section 98.2: Basic StringRequest using GET method	626
Section 98.3: Adding custom design time attributes to NetworkImageView	627
Section 98.4: Adding custom headers to your requests [e.g. for basic auth]	628
Section 98.5: Remote server authentication using StringRequest through POST method	629
Section 98.6: Cancel a request	631

第98.7节：请求JSON	631
第98.8节：使用JSONArray作为请求体	631
第98.9节：使用Volley发送JSON请求时服务器返回的布尔变量响应	632
第98.10节：处理Volley错误的辅助类	633
第99章：日期和时间选择器	635
第99.1节：日期选择对话框	635
第99.2节：Material日期选择器	635
第100章：Android中的本地化日期/时间	638
第100.1节：使用DateUtils.formatDateTime()自定义本地化日期格式	638
第100.2节：Android中的标准日期/时间格式化	638
第100.3节：完全自定义日期/时间	638
第101章：时间工具	639
第101.1节：检查是否在某一时间段内	639
第101.2节：将日期格式转换为毫秒	639
第101.3节：获取当前实时时间	640
第102章：应用内计费	641
第102.1节：可消耗的应用内购买	641
第102.2节：（第三方）应用内v3库	645
第103章：浮动操作按钮	647
第103.1节：如何将浮动操作按钮添加到布局中	647
第103.2节：在滑动时显示和隐藏浮动操作按钮	648
第103.3节：在滚动时显示和隐藏浮动操作按钮	650
第103.4节：FloatingActionButton的行为设置	652
第104章：触摸事件	653
第104.1节：如何区分子视图组和父视图组的触摸事件	653
第105章：处理触摸和运动事件	656
第105.1节：按钮	656
第105.2节：表面	657
第105.3节：在Surface中处理多点触控	658
第106章：在Android中检测摇晃事件	659
第106.1节：Android中摇晃检测器示例	659
第106.2节：使用地震摇晃检测	660
第107章：硬件按钮事件/意图（PTT、LWP等）	661
第107.1节：Sonim设备	661
第107.2节：RugGear设备	661
第108章：GreenRobot事件总线	662
第108.1节：传递简单事件	662
第108.2节：接收事件	663
第108.3节：发送事件	663
第109章：Otto事件总线	664
第109.1节：传递事件	664
第109.2节：接收事件	664
第110章：振动	666
第110.1节：振动入门	666
第110.2节：无限振动	666
第110.3节：振动模式	666
第110.4节：停止振动	667
第110.5节：振动一次	667

Section 98.7: Request JSON	631
Section 98.8: Use JSONArray as request body	631
Section 98.9: Boolean variable response from server with json request in volley	632
Section 98.10: Helper Class for Handling Volley Errors	633
Chapter 99: Date and Time Pickers	635
Section 99.1: Date Picker Dialog	635
Section 99.2: Material DatePicker	635
Chapter 100: Localized Date/Time in Android	638
Section 100.1: Custom localized date format with DateUtils.formatDateTime()	638
Section 100.2: Standard date/time formatting in Android	638
Section 100.3: Fully customized date/time	638
Chapter 101: Time Utils	639
Section 101.1: To check within a period	639
Section 101.2: Convert Date Format into Milliseconds	639
Section 101.3: GetCurrentRealTime	640
Chapter 102: In-app Billing	641
Section 102.1: Consumable In-app Purchases	641
Section 102.2: (Third party) In-App v3 Library	645
Chapter 103: FloatingActionButton	647
Section 103.1: How to add the FAB to the layout	647
Section 103.2: Show and Hide FloatingActionButton on Swipe	648
Section 103.3: Show and Hide FloatingActionButton on Scroll	650
Section 103.4: Setting behaviour of FloatingActionButton	652
Chapter 104: Touch Events	653
Section 104.1: How to vary between child and parent view group touch events	653
Chapter 105: Handling touch and motion events	656
Section 105.1: Buttons	656
Section 105.2: Surface	657
Section 105.3: Handling multitouch in a surface	658
Chapter 106: Detect Shake Event in Android	659
Section 106.1: Shake Detector in Android Example	659
Section 106.2: Using Seismic shake detection	660
Chapter 107: Hardware Button Events/Intents (PTT, LWP, etc.)	661
Section 107.1: Sonim Devices	661
Section 107.2: RugGear Devices	661
Chapter 108: GreenRobot EventBus	662
Section 108.1: Passing a Simple Event	662
Section 108.2: Receiving Events	663
Section 108.3: Sending Events	663
Chapter 109: Otto Event Bus	664
Section 109.1: Passing an event	664
Section 109.2: Receiving an event	664
Chapter 110: Vibration	666
Section 110.1: Getting Started with Vibration	666
Section 110.2: Vibrate Indefinitely	666
Section 110.3: Vibration Patterns	666
Section 110.4: Stop Vibrate	667
Section 110.5: Vibrate for one time	667

第111章：内容提供者	668	Chapter 111: ContentProvider	668
第111.1节：实现一个基本的内容提供者类	668	Section 111.1: Implementing a basic content provider class	668
第112章：Dagger 2	672	Chapter 112: Dagger 2	672
第112.1节：应用程序和活动注入的组件设置	672	Section 112.1: Component setup for Application and Activity injection	672
第112.2节：自定义作用域	673	Section 112.2: Custom Scopes	673
第112.3节：使用@Subcomponent代替@Component(dependencies={...})	674	Section 112.3: Using @Subcomponent instead of @Component(dependencies={...})	674
第112.4节：从多个模块创建组件	674	Section 112.4: Creating a component from multiple modules	674
第112.5节：如何在build.gradle中添加Dagger 2	675	Section 112.5: How to add Dagger 2 in build.gradle	675
第112.6节：构造函数注入	676	Section 112.6: Constructor Injection	676
第113章：Realm	678	Chapter 113: Realm	678
第113.1节：排序查询	678	Section 113.1: Sorted queries	678
第113.2节：在RxJava中使用Realm	678	Section 113.2: Using Realm with RxJava	678
第113.3节：基本用法	679	Section 113.3: Basic Usage	679
第113.4节：原始类型列表 (RealmList<Integer/String/...>)	682	Section 113.4: List of primitives (RealmList<Integer/String/...>)	682
第113.5节：异步查询	683	Section 113.5: Async queries	683
第113.6节：将Realm添加到你的项目中	683	Section 113.6: Adding Realm to your project	683
第113.7节：Realm模型	683	Section 113.7: Realm Models	683
第113.8节：try-with-resources语句	684	Section 113.8: try-with-resources	684
第114章：安卓版本	685	Chapter 114: Android Versions	685
第114.1节：运行时检查设备上的安卓版本	685	Section 114.1: Checking the Android Version on device at runtime	685
第115章：Wi-Fi连接	686	Chapter 115: Wi-Fi Connections	686
第115.1节：连接WEP加密	686	Section 115.1: Connect with WEP encryption	686
第115.2节：连接WPA2加密	686	Section 115.2: Connect with WPA2 encryption	686
第115.3节：扫描接入点	687	Section 115.3: Scan for access points	687
第116章：传感器管理器 (SensorManager)	689	Chapter 116: SensorManager	689
第116.1节：使用加速度计判断设备是否静止	689	Section 116.1: Decide if your device is static or not, using the accelerometer	689
第116.2节：获取传感器事件	689	Section 116.2: Retrieving sensor events	689
第116.3节：传感器转换到世界坐标系	690	Section 116.3: Sensor transformation to world coordinate system	690
第117章：进度条	692	Chapter 117: ProgressBar	692
第117.1节：材质线性进度条	692	Section 117.1: Material Linear ProgressBar	692
第117.2节：进度条着色	694	Section 117.2: Tinting ProgressBar	694
第117.3节：自定义进度条	694	Section 117.3: Customized progressbar	694
第117.4节：创建自定义进度对话框	696	Section 117.4: Creating Custom Progress Dialog	696
第117.5节：不确定进度条	698	Section 117.5: Indeterminate ProgressBar	698
第117.6节：确定进度条	699	Section 117.6: Determinate ProgressBar	699
第118章：自定义字体	701	Chapter 118: Custom Fonts	701
第118.1节：画布文本中的自定义字体	701	Section 118.1: Custom font in canvas text	701
第118.2节：在Android Q中使用字体	701	Section 118.2: Working with fonts in Android Q	701
第118.3节：为整个活动设置自定义字体	702	Section 118.3: Custom font to whole activity	702
第118.4节：在应用中使用自定义字体	702	Section 118.4: Putting a custom font in your app	702
第118.5节：初始化字体	702	Section 118.5: Initializing a font	702
第118.6节：在TextView中使用自定义字体	702	Section 118.6: Using a custom font in a TextView	702
第118.7节：通过xml在TextView上应用字体（无需Java代码）	703	Section 118.7: Apply font on TextView by xml (Not required Java code)	703
第118.8节：高效的字体加载	704	Section 118.8: Efficient Typeface loading	704
第119章：获取系统字体名称及使用字体	705	Chapter 119: Getting system font names and using the fonts	705
第119.1节：获取系统字体名称	705	Section 119.1: Getting system font names	705
第119.2节：将系统字体应用于TextView	705	Section 119.2: Applying a system font to a TextView	705
第120章：文本转语音 (TTS)	706	Chapter 120: Text to Speech(TTS)	706
第120.1节：文本转语音基础	706	Section 120.1: Text to Speech Base	706

第120.2节：跨API的文本转语音实现	707
第121章：下拉列表（Spinner）	711
第121.1节：基本下拉列表示例	711
第121.2节：向活动中添加下拉列表	712
第122章：数据加密/解密	714
第122.1节：使用密码以安全方式进行数据的AES加密	714
第123章：OkHttp	716
第123.1节：基本使用示例	716
第123.2节：设置OkHttp	716
第123.3节：日志拦截器	716
第123.4节：同步Get请求	717
第123.5节：异步Get请求	717
第123.6节：提交表单参数	718
第123.7节：发送多部分请求	718
第123.8节：重写响应	718
第124章：处理深度链接	720
第124.1节：检索查询参数	720
第124.2节：简单深度链接	720
第124.3节：单一域名上的多个路径	721
第124.4节：多个域名和多个路径	721
第124.5节：同一域名的http和https	722
第124.6节：使用pathPrefix	722
第125章：崩溃报告工具	723
第125.1节：Fabric - Crashlytics	723
第125.2节：使用Sherlock捕获崩溃	728
第125.3节：使用Fabric强制测试崩溃	729
第125.4节：使用ACRA进行崩溃报告	730
第126章：检查互联网连接	732
第126.1节：检查设备是否有互联网连接	732
第126.2节：如何检查安卓设备的网络强度？	732
第126.3节：如何检查网络强度	733
第127章：为安卓应用创建自己的库	736
第127.1节：创建可在Jitpack.io上使用的库	736
第127.2节：创建库项目	736
第127.3节：在项目中作为模块使用库	737
第128章：设备显示指标	738
第128.1节：获取屏幕像素尺寸	738
第128.2节：获取屏幕密度	738
第128.3节：像素(px)与密度无关像素(dp)转换公式	738
第129章：构建向后兼容的应用	739
第129.1节：如何处理已废弃的API	739
第130章：加载器	741
第130.1节：基础异步任务加载器	741
第130.2节：带缓存的异步任务加载器	742
第130.3节：重新加载	743
第130.4节：使用Bundle传递参数	744
第131章：ProGuard - 混淆和压缩你的代码	745
第131.1节：一些广泛使用的库的规则	745
第131.2节：在构建时移除跟踪日志（及其他）语句	747

Section 120.2: TextToSpeech implementation across the APIs	707
Chapter 121: Spinner	711
Section 121.1: Basic Spinner Example	711
Section 121.2: Adding a spinner to your activity	712
Chapter 122: Data Encryption/Decryption	714
Section 122.1: AES encryption of data using password in a secure way	714
Chapter 123: OkHttp	716
Section 123.1: Basic usage example	716
Section 123.2: Setting up OkHttp	716
Section 123.3: Logging interceptor	716
Section 123.4: Synchronous Get Call	717
Section 123.5: Asynchronous Get Call	717
Section 123.6: Posting form parameters	718
Section 123.7: Posting a multipart request	718
Section 123.8: Rewriting Responses	718
Chapter 124: Handling Deep Links	720
Section 124.1: Retrieving query parameters	720
Section 124.2: Simple deep link	720
Section 124.3: Multiple paths on a single domain	721
Section 124.4: Multiple domains and multiple paths	721
Section 124.5: Both http and https for the same domain	722
Section 124.6: Using pathPrefix	722
Chapter 125: Crash Reporting Tools	723
Section 125.1: Fabric - Crashlytics	723
Section 125.2: Capture crashes using Sherlock	728
Section 125.3: Force a Test Crash With Fabric	729
Section 125.4: Crash Reporting with ACRA	730
Chapter 126: Check Internet Connectivity	732
Section 126.1: Check if device has internet connectivity	732
Section 126.2: How to check network strength in android?	732
Section 126.3: How to check network strength	733
Chapter 127: Creating your own libraries for Android applications	736
Section 127.1: Create a library available on Jitpack.io	736
Section 127.2: Creating library project	736
Section 127.3: Using library in project as a module	737
Chapter 128: Device Display Metrics	738
Section 128.1: Get the screens pixel dimensions	738
Section 128.2: Get screen density	738
Section 128.3: Formula px to dp, dp to px conversion	738
Chapter 129: Building Backwards Compatible Apps	739
Section 129.1: How to handle deprecated API	739
Chapter 130: Loader	741
Section 130.1: Basic AsyncTaskLoader	741
Section 130.2: AsyncTaskLoader with cache	742
Section 130.3: Reloading	743
Section 130.4: Pass parameters using a Bundle	744
Chapter 131: ProGuard - Obfuscating and Shrinking your code	745
Section 131.1: Rules for some of the widely used Libraries	745
Section 131.2: Remove trace logging (and other) statements at build time	747

第131.3节：保护你的代码免受黑客攻击	747
第131.4节：为你的构建启用ProGuard	748
第131.5节：使用自定义混淆配置文件启用ProGuard	748
第132章：类型定义注解：@IntDef, @StringDef	750
第132.1节：IntDef注解	750
第132.2节：结合常量与标志	750
第133章：截取屏幕截图	752
第133.1节：截取特定视图的屏幕截图	752
第133.2节：通过Android Studio截取屏幕截图	752
第133.3节：通过ADB截取屏幕截图并直接保存到电脑	753
第133.4节：通过Android设备监视器截取屏幕截图	753
第133.5节：通过ADB截取屏幕截图	754
第134章：MVP架构	755
第134.1节：模型视图主持人（MVP）模式中的登录示例	755
第134.2节：MVP中的简单登录示例	758
第135章：方向变化	765
第135.1节：保存和恢复活动状态	765
第135.2节：保留碎片	765
第135.3节：手动管理配置更改	766
第135.4节：处理异步任务（AsyncTask）	767
第135.5节：通过编程控制锁屏旋转	768
第135.6节：保存和恢复Fragment状态	769
第136章：Xposed	771
第136.1节：创建Xposed模块	771
第136.2节：Hook方法	771
第137章：PackageManager	773
第137.1节：获取应用版本	773
第137.2节：版本名称和版本代码	773
第137.3节：安装时间和更新时间	773
第137.4节：使用PackageManager的实用方法	774
第138章：手势检测	776
第138.1节：滑动检测	776
第138.2节：基本手势检测	777
第139章：休眠模式	779
第139.1节：以编程方式将Android应用程序加入白名单	779
第139.2节：排除应用使用休眠模式	779
第140章：颜色	780
第140.1节：颜色操作	780
第141章：键盘	781
第141.1节：注册键盘打开和关闭的回调	781
第141.2节：当用户点击屏幕其他任何地方时隐藏键盘	781
第142章：RenderScript	783
第142.1节：入门	783
第142.2节：模糊视图	789
第142.3节：模糊图像	791
第143章：Fresco	794
第143.1节：Fresco入门	794
第143.2节：使用OkHttp 3与Fresco	795

Section 131.3: Protecting your code from hackers	747
Section 131.4: Enable ProGuard for your build	748
Section 131.5: Enabling ProGuard with a custom obfuscation configuration file	748
Chapter 132: Typedef Annotations: @IntDef, @StringDef	750
Section 132.1: IntDef Annotations	750
Section 132.2: Combining constants with flags	750
Chapter 133: Capturing Screenshots	752
Section 133.1: Taking a screenshot of a particular view	752
Section 133.2: Capturing Screenshot via Android Studio	752
Section 133.3: Capturing Screenshot via ADB and saving directly in your PC	753
Section 133.4: Capturing Screenshot via Android Device Monitor	753
Section 133.5: Capturing Screenshot via ADB	754
Chapter 134: MVP Architecture	755
Section 134.1: Login example in the Model View Presenter (MVP) pattern	755
Section 134.2: Simple Login Example in MVP	758
Chapter 135: Orientation Changes	765
Section 135.1: Saving and Restoring Activity State	765
Section 135.2: Retaining Fragments	765
Section 135.3: Manually Managing Configuration Changes	766
Section 135.4: Handling AsyncTask	767
Section 135.5: Lock Screen's rotation programmatically	768
Section 135.6: Saving and Restoring Fragment State	769
Chapter 136: Xposed	771
Section 136.1: Creating a Xposed Module	771
Section 136.2: Hooking a method	771
Chapter 137: PackageManager	773
Section 137.1: Retrieve application version	773
Section 137.2: Version name and version code	773
Section 137.3: Install time and update time	773
Section 137.4: Utility method using PackageManager	774
Chapter 138: Gesture Detection	776
Section 138.1: Swipe Detection	776
Section 138.2: Basic Gesture Detection	777
Chapter 139: Doze Mode	779
Section 139.1: Whitelisting an Android application programmatically	779
Section 139.2: Exclude app from using doze mode	779
Chapter 140: Colors	780
Section 140.1: Color Manipulation	780
Chapter 141: Keyboard	781
Section 141.1: Register a callback for keyboard open and close	781
Section 141.2: Hide keyboard when user taps anywhere else on the screen	781
Chapter 142: RenderScript	783
Section 142.1: Getting Started	783
Section 142.2: Blur a View	789
Section 142.3: Blur an image	791
Chapter 143: Fresco	794
Section 143.1: Getting Started with Fresco	794
Section 143.2: Using OkHttp 3 with Fresco	795

第143.3节：使用DraweeController进行Fresco的JPEG流式传输	795
第144章：下拉刷新	796
第144.1节：如何向应用添加下拉刷新	796
第144.2节：RecyclerView的下拉刷新	796
第145章：创建启动画面	798
第145.1节：带动画的启动画面	798
第145.2节：一个基本的启动画面	799
第146章：IntentService	802
第146.1节：创建IntentService	802
第146.2节：基本的IntentService示例	802
第146.3节：示例Intent服务	803
第147章：隐式Intent	805
第147.1节：隐式和显式Intent	805
第147.2节：隐式意图	805
第148章：发布到Play商店	806
第148.1节：最简应用提交指南	806
第149章：通用图片加载器	808
第149.1节：基本用法	808
第149.2节：初始化通用图片加载器	808
第150章：图片压缩	809
第150.1节：如何在不改变大小的情况下压缩图像	809
第151章：9-切片图像	812
第151.1节：基本圆角	812
第151.2节：可选填充线	812
第151.3节：基本加载动画	813
第152章：电子邮件验证	814
第152.1节：电子邮件地址验证	814
第152.2节：使用模式的电子邮件地址验证	814
第153章：底部面板	815
第153.1节：快速设置	815
第153.2节：类似谷歌地图的BottomSheetBehavior	815
第153.3节：使用BottomSheetDialog的模态底部面板	822
第153.4节：使用BottomSheetDialogFragment的模态底部面板	822
第153.5节：持久性底部面板	822
第153.6节：默认以展开模式打开BottomSheet DialogFragment	823
第154章：编辑文本框	825
第154.1节：使用EditTexts	825
第154.2节：自定义InputType	827
第154.3节：自定义编辑文本中的图标或按钮及其操作和点击监听器	827
第154.4节：隐藏软键盘	829
第154.5节：`inputtype`属性	830
第155章：语音转文本	832
第155.1节：使用默认Google提示对话框的语音转文本	832
第155.2节：无对话框的语音转文本	833
第156章：使用ADB安装应用	835
第156.1节：卸载应用	835
第156.2节：安装目录中的所有apk文件	835
第156.3节：安装应用	835
Section 143.3: JPEG Streaming with Fresco using DraweeController	795
Chapter 144: Swipe to Refresh	796
Section 144.1: How to add Swipe-to-Refresh To your app	796
Section 144.2: Swipe To Refresh with RecyclerView	796
Chapter 145: Creating Splash screen	798
Section 145.1: Splash screen with animation	798
Section 145.2: A basic splash screen	799
Chapter 146: IntentService	802
Section 146.1: Creating an IntentService	802
Section 146.2: Basic IntentService Example	802
Section 146.3: Sample Intent Service	803
Chapter 147: Implicit Intents	805
Section 147.1: Implicit and Explicit Intents	805
Section 147.2: Implicit Intents	805
Chapter 148: Publish to Play Store	806
Section 148.1: Minimal app submission guide	806
Chapter 149: Universal Image Loader	808
Section 149.1: Basic usage	808
Section 149.2: Initialize Universal Image Loader	808
Chapter 150: Image Compression	809
Section 150.1: How to compress image without size change	809
Chapter 151: 9-Patch Images	812
Section 151.1: Basic rounded corners	812
Section 151.2: Optional padding lines	812
Section 151.3: Basic spinner	813
Chapter 152: Email Validation	814
Section 152.1: Email address validation	814
Section 152.2: Email Address validation with using Patterns	814
Chapter 153: Bottom Sheets	815
Section 153.1: Quick Setup	815
Section 153.2: BottomSheetBehavior like Google maps	815
Section 153.3: Modal bottom sheets with BottomSheetDialog	822
Section 153.4: Modal bottom sheets with BottomSheetDialogFragment	822
Section 153.5: Persistent Bottom Sheets	822
Section 153.6: Open BottomSheet DialogFragment in Expanded mode by default	823
Chapter 154: EditText	825
Section 154.1: Working with EditTexts	825
Section 154.2: Customizing the InputType	827
Section 154.3: Icon or button inside Custom Edit Text and its action and click listeners	827
Section 154.4: Hiding SoftKeyboard	829
Section 154.5: `inputtype` attribute	830
Chapter 155: Speech to Text Conversion	832
Section 155.1: Speech to Text With Default Google Prompt Dialog	832
Section 155.2: Speech to Text without Dialog	833
Chapter 156: Installing apps with ADB	835
Section 156.1: Uninstall an app	835
Section 156.2: Install all apk file in directory	835
Section 156.3: Install an app	835

第157章：倒计时定时器	836
第157.1节：创建一个简单的倒计时器	836
第157.2节：一个更复杂的示例	836
第158章：条形码和二维码读取	838
第158.1节：使用基于Zxing的QRCodeReaderView	838
第159章：Android PayPal网关集成	840
第159.1节：在你的Android代码中设置PayPal	840
第160章：可绘制对象	842
第160.1节：自定义Drawable	842
第160.2节：为Drawable着色	843
第160.3节：圆形视图	844
第160.4节：制作圆角视图	844
第161章：TransitionDrawable	846
第161.1节：使用TransitionDrawable动画视图背景颜色（切换颜色）	846
第161.2节：在两张图片之间添加过渡或交叉淡入淡出效果	846
第162章：矢量图形	848
第162.1节：将SVG文件导入为VectorDrawable	848
第162.2节：VectorDrawable使用示例	850
第162.3节：VectorDrawable XML示例	851
第163章：VectorDrawable和AnimatedVectorDrawable	852
第163.1节：基础VectorDrawable	852
第163.2节： <code><group></code> 标签	852
第163.3节：基础 AnimatedVectorDrawable	853
第163.4节：使用描边	854
第163.5节：使用 <code><clip-path></code>	856
第163.6节：通过 AppCompat 实现矢量兼容性	856
第164章：在 Android 中使用 Cling 库进行端口映射	858
第164.1节：映射NAT端口	858
第164.2节：为您的安卓项目添加Cling支持	858
第165章：创建覆盖（始终置顶）窗口	860
第165.1节：弹出覆盖层	860
第165.2节：在安卓6.0及以上版本授予SYSTEM_ALERT_WINDOW权限	860
第166章：ExoPlayer	862
第166.1节：将ExoPlayer添加到项目中	862
第166.2节：使用ExoPlayer	862
第166.3节：使用标准TrackRenderer实现播放视频和音频的主要步骤	863
第167章：XMPP注册登录和聊天简单示例	864
第167.1节：XMPP注册登录和聊天基础示例	864
第168章：Android身份验证器	873
第168.1节：基础账户身份验证服务	873
第169章：音频管理器	876
第169.1节：请求临时音频焦点	876
第169.2节：请求音频焦点	876
第170章：AudioTrack	877
第170.1节：生成特定频率的音调	877
第171章：作业调度	878
第171.1节：基本用法	878

Chapter 157: Count Down Timer	836
Section 157.1: Creating a simple countdown timer	836
Section 157.2: A More Complex Example	836
Chapter 158: Barcode and QR code reading	838
Section 158.1: Using QRCodeReaderView (based on Zxing)	838
Chapter 159: Android PayPal Gateway Integration	840
Section 159.1: Setup PayPal in your android code	840
Chapter 160: Drawables	842
Section 160.1: Custom Drawable	842
Section 160.2: Tint a drawable	843
Section 160.3: Circular View	844
Section 160.4: Make View with rounded corners	844
Chapter 161: TransitionDrawable	846
Section 161.1: Animate views background color (switch-color) with TransitionDrawable	846
Section 161.2: Add transition or Cross-fade between two images	846
Chapter 162: Vector Drawables	848
Section 162.1: Importing SVG file as VectorDrawable	848
Section 162.2: VectorDrawable Usage Example	850
Section 162.3: VectorDrawable xml example	851
Chapter 163: VectorDrawable and AnimatedVectorDrawable	852
Section 163.1: Basic VectorDrawable	852
Section 163.2: <code><group></code> tags	852
Section 163.3: Basic AnimatedVectorDrawable	853
Section 163.4: Using Strokes	854
Section 163.5: Using <code><clip-path></code>	856
Section 163.6: Vector compatibility through AppCompat	856
Chapter 164: Port Mapping using Cling library in Android	858
Section 164.1: Mapping a NAT port	858
Section 164.2: Adding Cling Support to your Android Project	858
Chapter 165: Creating Overlay (always-on-top) Windows	860
Section 165.1: Popup overlay	860
Section 165.2: Granting SYSTEM_ALERT_WINDOW Permission on android 6.0 and above	860
Chapter 166: ExoPlayer	862
Section 166.1: Add ExoPlayer to the project	862
Section 166.2: Using ExoPlayer	862
Section 166.3: Main steps to play video & audio using the standard TrackRenderer implementations	863
Chapter 167: XMPP register login and chat simple example	864
Section 167.1: XMPP register login and chat basic example	864
Chapter 168: Android Authenticator	873
Section 168.1: Basic Account Authenticator Service	873
Chapter 169: AudioManager	876
Section 169.1: Requesting Transient Audio Focus	876
Section 169.2: Requesting Audio Focus	876
Chapter 170: AudioTrack	877
Section 170.1: Generate tone of a specific frequency	877
Chapter 171: Job Scheduling	878
Section 171.1: Basic usage	878

第172章：账户与AccountManager	880
第172.1节：理解自定义账户/认证	880
第173章：将OpenCV集成到Android Studio	882
第173.1节：说明	882
第174章：MVVM（架构）	890
第174.1节：使用DataBinding库的MVVM示例	890
第175章：Android中的ORMLite	897
第175.1节：基于SQLite的Android OrmLite示例	897
第176章：使用RxJava的Retrofit2	901
第176.1节：Retrofit2与RxJava	901
第176.2节：嵌套请求示例：多个请求，合并结果	902
第176.3节：使用Retrofit和RxJava异步获取数据	903
第177章：ShortcutManager	906
第177.1节：动态启动器快捷方式	906
第178章：LruCache	907
第178.1节：向缓存添加位图（资源）	907
第178.2节：初始化缓存	907
第178.3节：从缓存获取位图（资源）	907
第179章：Android项目的Jenkins持续集成设置	908
第179.1节：设置Jenkins以支持Android的逐步方法	908
第180章：fastlane	912
第180.1节：Fastfile通道，用于构建并安装给定构建类型的所有版本到设备上	912
第180.2节：使用Fastfile构建并通过Crashlytics上传多个版本到Beta	912
第181章：为自定义RangeSeekBar定义步进值（增量）	915
第181.1节：定义步进值为7	915
第182章：OpenGL ES 2.0+入门	916
第182.1节：设置GLSurfaceView和OpenGL ES 2.0+	916
第182.2节：从资源文件编译和链接GLSL-ES着色器	916
第183章：检查数据连接	919
第183.1节：检查数据连接	919
第183.2节：使用ConnectivityManager检查连接	919
第183.3节：使用网络意图在允许数据时执行任务	919
第184章：Android上的Java	920
第184.1节：使用Retrolambda的Java 8功能子集	920
第185章：Android Java本地接口（JNI）	922
第185.1节：如何通过JNI接口调用本地库中的函数	922
第185.2节：如何从本地代码调用Java方法	922
第185.3节：JNI层中的实用方法	923
第186章：通知渠道 Android O	925
第186.1节：通知渠道	925
第187章：Robolectric	931
第187.1节：Robolectric 测试	931
第187.2节：配置	931
第188章：Moshi	933
第188.1节：将JSON转换为Java	933
第188.2节：将Java对象序列化为JSON	933
第188.3节：内置类型适配器	933

Chapter 172: Accounts and AccountManager	880
Section 172.1: Understanding custom accounts/authentication	880
Chapter 173: Integrate OpenCV into Android Studio	882
Section 173.1: Instructions	882
Chapter 174: MVVM (Architecture)	890
Section 174.1: MVVM Example using DataBinding Library	890
Chapter 175: ORMLite in android	897
Section 175.1: Android OrmLite over SQLite example	897
Chapter 176: Retrofit2 with RxJava	901
Section 176.1: Retrofit2 with RxJava	901
Section 176.2: Nested requests example: multiple requests, combine results	902
Section 176.3: Retrofit with RxJava to fetch data asynchronously	903
Chapter 177: ShortcutManager	906
Section 177.1: Dynamic Launcher Shortcuts	906
Chapter 178: LruCache	907
Section 178.1: Adding a Bitmap(Resource) to the cache	907
Section 178.2: Initialising the cache	907
Section 178.3: Getting a Bitmap(Resource) from the cache	907
Chapter 179: Jenkins CI setup for Android Projects	908
Section 179.1: Step by step approach to set up Jenkins for Android	908
Chapter 180: fastlane	912
Section 180.1: Fastfile lane to build and install all flavors for given build type to a device	912
Section 180.2: Fastfile to build and upload multiple flavors to Beta by Crashlytics	912
Chapter 181: Define step value (increment) for custom RangeSeekBar	915
Section 181.1: Define a step value of 7	915
Chapter 182: Getting started with OpenGL ES 2.0+	916
Section 182.1: Setting up GLSurfaceView and OpenGL ES 2.0+	916
Section 182.2: Compiling and Linking GLSL-ES Shaders from asset file	916
Chapter 183: Check Data Connection	919
Section 183.1: Check data connection	919
Section 183.2: Check connection using ConnectivityManager	919
Section 183.3: Use network intents to perform tasks while data is allowed	919
Chapter 184: Java on Android	920
Section 184.1: Java 8 features subset with Retrolambda	920
Chapter 185: Android Java Native Interface (JNI)	922
Section 185.1: How to call functions in a native library via the JNI interface	922
Section 185.2: How to call a Java method from native code	922
Section 185.3: Utility method in JNI layer	923
Chapter 186: Notification Channel Android O	925
Section 186.1: Notification Channel	925
Chapter 187: Robolectric	931
Section 187.1: Robolectric test	931
Section 187.2: Configuration	931
Chapter 188: Moshi	933
Section 188.1: JSON into Java	933
Section 188.2: serialize Java objects as JSON	933
Section 188.3: Built in Type Adapters	933

第189章：严格模式策略：一种在编译时捕捉错误的工具	935
第189.1节：以下代码片段用于设置线程策略的严格模式。此代码应设置 在我们应用程序的入口点处	935
第189.2节：以下代码处理内存泄漏，例如检测SQLite的finalize方法是否被调用 或未被调用	935
第190章：国际化和本地化 (I18N 和 L10N)	936
第190.1节：本地化规划：在清单中启用从右到左 (RTL) 支持	936
第190.2节：本地化规划：在布局中添加从右到左 (RTL) 支持	936
第190.3节：本地化规划：测试从右到左 (RTL) 布局	937
第190.4节：本地化编码：创建默认字符串和资源	937
第190.5节：本地化编码：提供替代字符串	938
第190.6节：本地化编码：提供备用布局	939
第191章：在安卓项目中快速设置Retrolambda的方法	940
第191.1节：设置及使用示例	940
第192章：如何使用SparseArray	942
第192.1节：使用SparseArray的基本示例	942
第193章：共享元素过渡	944
第193.1节：两个片段之间的共享元素过渡	944
第194章：Android Things	947
第194.1节：控制舵机	947
第195章：库 Dagger 2：应用中的依赖注入	949
第195.1节：创建 @Module 类和 @Singleton 注解对象	949
第195.2节：在依赖对象中请求依赖	949
第195.3节：使用 @Inject 连接 @Modules	949
第195.4节：使用 @Component 接口获取对象	950
第196章：JCodec	951
第196.1节：入门	951
第196.2节：从影片中获取帧	951
第197章：使用模式格式化电话号码	952
第197.1节：模式 + 1 (786) 1234 5678	952
第198章：绘画	953
第198.1节：创建绘画	953
第198.2节：为文本设置油漆	953
第198.3节：设置绘制形状的画笔	954
第198.4节：设置标志	954
第199章：什么是ProGuard？它在安卓中的作用是什么？	955
第199.1节：使用ProGuard压缩代码和资源	955
第200章：创建安卓自定义ROM	957
第200.1节：准备构建环境！	957
第201章：安卓的Genymotion	958
第201.1节：安装Genymotion免费版	958
第201.2节：Genymotion上的谷歌框架	959
第202章：ConstraintSet	960
第202.1节：通过编程方式使用ConstraintLayout的ConstraintSet	960
第203章：CleverTap	961
第203.1节：设置调试级别	961
第203.2节：获取SDK实例以记录事件	961
第204章：发布库到Maven仓库	962

Chapter 189: Strict Mode Policy : A tool to catch the bug in the Compile Time.	935
Section 189.1: The below Code Snippet is to setup the StrictMode for Thread Policies. This Code is to be set at the entry points to our application	935
Section 189.2: The below code deals with leaks of memory, like it detects when in SQLite finalize is called or not	935
Chapter 190: Internationalization and localization (I18N and L10N)	936
Section 190.1: Planning for localization : enable RTL support in Manifest	936
Section 190.2: Planning for localization : Add RTL support in Layouts	936
Section 190.3: Planning for localization : Test layouts for RTL	937
Section 190.4: Coding for Localization : Creating default strings and resources	937
Section 190.5: Coding for localization : Providing alternative strings	938
Section 190.6: Coding for localization : Providing alternate layouts	939
Chapter 191: Fast way to setup Retrolambda on an android project.	940
Section 191.1: Setup and example how to use:	940
Chapter 192: How to use SparseArray	942
Section 192.1: Basic example using SparseArray	942
Chapter 193: Shared Element Transitions	944
Section 193.1: Shared Element Transition between two Fragments	944
Chapter 194: Android Things	947
Section 194.1: Controlling a Servo Motor	947
Chapter 195: Library Dagger 2: Dependency Injection in Applications	949
Section 195.1: Create @Module Class and @Singleton annotation for Object	949
Section 195.2: Request Dependencies in Dependent Objects	949
Section 195.3: Connecting @Modules with @Inject	949
Section 195.4: Using @Component Interface to Obtain Objects	950
Chapter 196: JCodec	951
Section 196.1: Getting Started	951
Section 196.2: Getting frame from movie	951
Chapter 197: Formatting phone numbers with pattern.	952
Section 197.1: Patterns +1(786) 1234 5678	952
Chapter 198: Paint	953
Section 198.1: Creating a Paint	953
Section 198.2: Setting up Paint for text	953
Section 198.3: Setting up Paint for drawing shapes	954
Section 198.4: Setting flags	954
Chapter 199: What is ProGuard? What is use in Android?	955
Section 199.1: Shrink your code and resources with proguard	955
Chapter 200: Create Android Custom ROMs	957
Section 200.1: Making Your Machine Ready for Building!	957
Chapter 201: Genymotion for android	958
Section 201.1: Installing Genymotion, the free version	958
Section 201.2: Google framework on Genymotion	959
Chapter 202: ConstraintSet	960
Section 202.1: ConstraintSet with ConstraintLayout Programmatically	960
Chapter 203: CleverTap	961
Section 203.1: Setting the debug level	961
Section 203.2: Get an instance of the SDK to record events	961
Chapter 204: Publish a library to Maven Repositories	962

第204.1节：发布.aar文件到Maven	962
第205章：adb shell	964
第205.1节：授予和撤销API 23及以上权限	964
第205.2节：通过ADB向Android设备发送文本、按键和触摸事件	964
第205.3节：列出包	966
第205.4节：录制显示屏	966
第205.5节：打开开发者选项	967
第205.6节：通过adb设置日期/时间	967
第205.7节：生成“启动完成”广播	968
第205.8节：打印应用数据	968
第205.9节：使用chmod命令更改文件权限	968
第205.10节：查看外部/二级存储内容	969
第205.11节：在Android设备中终止进程	969
第206章：Ping ICMP	971
第206.1节：执行单次Ping	971
第207章：AIDL	972
第207.1节：AIDL服务	972
第208章：Android游戏开发	974
第208.1节：使用Canvas和SurfaceView的游戏	974
第209章：使用Kotlin进行Android编程	980
第209.1节：安装Kotlin插件	980
第209.2节：为现有Gradle项目配置Kotlin	981
第209.3节：创建新的Kotlin活动	982
第209.4节：将现有Java代码转换为Kotlin	983
第209.5节：启动新的活动	983
第210章：VirtualBox中的Android-x86	984
第210.1节：支持SD卡的虚拟硬盘设置	984
第210.2节：隔断中的安装	986
第210.3节：虚拟机设置	988
第211章：Leakcanary（内存泄漏检测工具）	989
第211.1节：在安卓应用中实现Leak Canary	989
第212章：Okio（高效I/O库）	990
第212.1节：下载 / 实现	990
第212.2节：PNG解码器	990
第212.3节：字节串和缓冲区	990
第213章：蓝牙低功耗	992
第213.1节：查找BLE设备	992
第213.2节：连接到GATT服务器	992
第213.3节：写入和读取特征值	993
第213.4节：订阅来自GATT服务器的通知	994
第213.5节：BLE设备的广告	994
第213.6节：使用Gatt服务器	995
第214章：Looper	997
第214.1节：创建一个简单的Looper线程	997
第214.2节：使用Handler线程运行循环	997
第215章：注解处理器	998
第215.1节：@NotNull注解	998
第215.2节：注释类型	998
第215.3节：创建和使用自定义注释	998

Section 204.1: Publish .aar file to Maven	962
Chapter 205: adb shell	964
Section 205.1: Granting & revoking API 23+ permissions	964
Section 205.2: Send text, key pressed and touch events to Android Device via ADB	964
Section 205.3: List packages	966
Section 205.4: Recording the display	966
Section 205.5: Open Developer Options	967
Section 205.6: Set Date/Time via adb	967
Section 205.7: Generating a "Boot Complete" broadcast	968
Section 205.8: Print application data	968
Section 205.9: Changing file permissions using chmod command	968
Section 205.10: View external/secondary storage content	969
Section 205.11: kill a process inside an Android device	969
Chapter 206: Ping ICMP	971
Section 206.1: Performs a single Ping	971
Chapter 207: AIDL	972
Section 207.1: AIDL Service	972
Chapter 208: Android game development	974
Section 208.1: Game using Canvas and SurfaceView	974
Chapter 209: Android programming with Kotlin	980
Section 209.1: Installing the Kotlin plugin	980
Section 209.2: Configuring an existing Gradle project with Kotlin	981
Section 209.3: Creating a new Kotlin Activity	982
Section 209.4: Converting existing Java code to Kotlin	983
Section 209.5: Starting a new Activity	983
Chapter 210: Android-x86 in VirtualBox	984
Section 210.1: Virtual hard drive Setup for SDCARD Support	984
Section 210.2: Installation in partition	986
Section 210.3: Virtual Machine setup	988
Chapter 211: Leakcanary	989
Section 211.1: Implementing a Leak Canary in Android Application	989
Chapter 212: Okio	990
Section 212.1: Download / Implement	990
Section 212.2: PNG decoder	990
Section 212.3: ByteString and Buffers	990
Chapter 213: Bluetooth Low Energy	992
Section 213.1: Finding BLE Devices	992
Section 213.2: Connecting to a GATT Server	992
Section 213.3: Writing and Reading from Characteristics	993
Section 213.4: Subscribing to Notifications from the Gatt Server	994
Section 213.5: Advertising a BLE Device	994
Section 213.6: Using a Gatt Server	995
Chapter 214: Looper	997
Section 214.1: Create a simple LooperThread	997
Section 214.2: Run a loop with a HandlerThread	997
Chapter 215: Annotation Processor	998
Section 215.1: @NotNull Annotation	998
Section 215.2: Types of Annotations	998
Section 215.3: Creating and Using Custom Annotations	998

216章：具有定期数据同步功能的SyncAdapter	1000	Chapter 216: SyncAdapter with periodically do sync of data	1000
216.1节：每分钟从服务器请求值的同步适配器	1000	Section 216.1: Sync adapter with every min requesting value from server	1000
217章：Fastjson	1010	Chapter 217: Fastjson	1010
217.1节：使用Fastjson解析JSON	1010	Section 217.1: Parsing JSON with Fastjson	1010
217.2节：将Map类型数据转换为JSON字符串	1011	Section 217.2: Convert the data of type Map to JSON String	1011
第218章：Android中使用org.json处理JSON	1013	Chapter 218: JSON in Android with org.json	1013
第218.1节：创建一个简单的JSON对象	1013	Section 218.1: Creating a simple JSON object	1013
第218.2节：创建一个带有null值的JSON字符串	1013	Section 218.2: Create a JSON String with null value	1013
第218.3节：向JSONObject添加JSONArray	1013	Section 218.3: Add JSONArray to JSONObject	1013
第218.4节：解析简单的JSON对象	1014	Section 218.4: Parse simple JSON object	1014
第218.5节：检查JSON字段的存在性	1015	Section 218.5: Check for the existence of fields on JSON	1015
第218.6节：创建嵌套的JSON对象	1015	Section 218.6: Create nested JSON object	1015
第218.7节：更新JSON中的元素	1016	Section 218.7: Updating the elements in the JSON	1016
第218.8节：使用JsonReader从流中读取JSON	1016	Section 218.8: Using JsonReader to read JSON from a stream	1016
第218.9节：解析JSON时处理空字符串	1018	Section 218.9: Working with null-string when parsing json	1018
第218.10节：处理JSON响应中的动态键	1019	Section 218.10: Handling dynamic key for JSON response	1019
第219章：Gson	1021	Chapter 219: Gson	1021
第219.1节：使用Gson解析JSON	1021	Section 219.1: Parsing JSON with Gson	1021
第219.2节：向Gson添加自定义转换器	1023	Section 219.2: Adding a custom Converter to Gson	1023
第219.3节：使用Gson解析List<String>	1023	Section 219.3: Parsing a List<String> with Gson	1023
第219.4节：将Gson添加到你的项目中	1024	Section 219.4: Adding Gson to your project	1024
第219.5节：使用Gson解析JSON到泛型类对象	1024	Section 219.5: Parsing JSON to Generic Class Object with Gson	1024
第219.6节：在继承中使用Gson	1025	Section 219.6: Using Gson with inheritance	1025
第219.7节：使用Gson将JSON属性解析为枚举	1027	Section 219.7: Parsing JSON property to enum with Gson	1027
第219.8节：使用Gson从磁盘加载JSON文件	1027	Section 219.8: Using Gson to load a JSON file from disk	1027
第219.9节：使用Gson作为Retrofit的序列化器	1027	Section 219.9: Using Gson as serializer with Retrofit	1027
第219.10节：使用Gson解析json数组到通用类	1028	Section 219.10: Parsing json array to generic class using Gson	1028
第219.11节：使用Gson自定义JSON反序列化器	1029	Section 219.11: Custom JSON Deserializer using Gson	1029
第219.12节：使用AutoValue和Gson进行JSON序列化/反序列化	1030	Section 219.12: JSON Serialization/Deserialization with AutoValue and Gson	1030
第220章：Android架构组件	1032	Chapter 220: Android Architecture Components	1032
第220.1节：在AppCompatActivity中使用生命周期	1032	Section 220.1: Using Lifecycle in AppCompatActivity	1032
第220.2节：添加架构组件	1032	Section 220.2: Add Architecture Components	1032
第220.3节：带有LiveData转换的ViewModel	1033	Section 220.3: ViewModel with LiveData transformations	1033
第220.4节：Room持久化	1034	Section 220.4: Room persistence	1034
第220.5节：自定义LiveData	1036	Section 220.5: Custom LiveData	1036
第220.6节：自定义生命周期感知组件	1036	Section 220.6: Custom Lifecycle-aware component	1036
第221章：Jackson	1038	Chapter 221: Jackson	1038
第221.1节：完整的数据绑定示例	1038	Section 221.1: Full Data Binding Example	1038
第222章：智能卡	1040	Chapter 222: Smartcard	1040
第222.1节：智能卡发送与接收	1040	Section 222.1: Smart card send and receive	1040
第223章：安全性	1042	Chapter 223: Security	1042
第223.1节：验证应用签名 - 防篡改检测	1042	Section 223.1: Verifying App Signature - Tamper Detection	1042
第224章：如何安全存储密码	1043	Chapter 224: How to store passwords securely	1043
第224.1节：使用AES进行加盐密码加密	1043	Section 224.1: Using AES for salted password encryption	1043
第225章：安全的SharedPreferences	1046	Chapter 225: Secure SharedPreferences	1046
第225.1节：确保共享偏好	1046	Section 225.1: Securing a Shared Preference	1046
第226章：安全的SharedPreferences	1047	Chapter 226: Secure SharedPreferences	1047
第226.1节：保护Shared Preference	1047	Section 226.1: Securing a Shared Preference	1047

第227章：SQLite	1048	Chapter 227: SQLite	1048
第227.1节：onUpgrade()方法	1048	Section 227.1: onUpgrade() method	1048
第227.2节：从Cursor读取数据	1048	Section 227.2: Reading data from a Cursor	1048
第227.3节：使用SQLiteOpenHelper类	1050	Section 227.3: Using the SQLiteOpenHelper class	1050
第227.4节：向数据库插入数据	1051	Section 227.4: Insert data into database	1051
第227.5节：批量插入	1051	Section 227.5: Bulk insert	1051
第227.6节：在Android中为SQLite创建合同、辅助类和提供者	1052	Section 227.6: Create a Contract, Helper and Provider for SQLite in Android	1052
第227.7节：从表中删除行	1056	Section 227.7: Delete row(s) from the table	1056
第227.8节：更新表中的行	1057	Section 227.8: Updating a row in a table	1057
第227.9节：执行事务	1057	Section 227.9: Performing a Transaction	1057
第227.10节：从assets文件夹创建数据库	1058	Section 227.10: Create Database from assets folder	1058
第227.11节：将图像存储到SQLite	1060	Section 227.11: Store image into SQLite	1060
第227.12节：导出和导入数据库	1062	Section 227.12: Exporting and importing a database	1062
第228章：使用ContentValues类访问SQLite数据库	1064	Chapter 228: Accessing SQLite databases using the ContentValues class	1064
第228.1节：在SQLite数据库中插入和更新行	1064	Section 228.1: Inserting and updating rows in a SQLite database	1064
第229章：Firebase	1065	Chapter 229: Firebase	1065
第229.1节：将Firebase添加到您的安卓项目	1065	Section 229.1: Add Firebase to Your Android Project	1065
第229.2节：更新Firebase用户的邮箱	1066	Section 229.2: Updating a Firebase users's email	1066
第229.3节：创建Firebase用户	1067	Section 229.3: Create a Firebase user	1067
第229.4节：更改密码	1068	Section 229.4: Change Password	1068
第229.5节：Firebase云消息服务	1069	Section 229.5: Firebase Cloud Messaging	1069
第229.6节：Firebase存储操作	1071	Section 229.6: Firebase Storage Operations	1071
第229.7节：Firebase实时数据库：如何设置/获取数据	1077	Section 229.7: Firebase Realtime Database: how to set/get data	1077
第229.8节：基于FCM通知的演示	1078	Section 229.8: Demo of FCM based notifications	1078
第229.9节：使用邮箱和密码登录Firebase用户	1088	Section 229.9: Sign In Firebase user with email and password	1088
第229.10节：发送Firebase密码重置邮件	1089	Section 229.10: Send Firebase password reset email	1089
第229.11节：重新认证Firebase用户	1091	Section 229.11: Re-Authenticate Firebase user	1091
第229.12节：Firebase 登出	1092	Section 229.12: Firebase Sign Out	1092
第230章：Firebase 云消息传递	1093	Chapter 230: Firebase Cloud Messaging	1093
第230.1节：在安卓上设置 Firebase 云消息传递客户端应用	1093	Section 230.1: Set Up a Firebase Cloud Messaging Client App on Android	1093
第230.2节：接收消息	1093	Section 230.2: Receive Messages	1093
我在应用中实现的用于推送图片、消息以及链接的代码 在你的 WebView 中打开	1094	Section 230.3: This code that i have implemnted in my app for pushing image,message and also link for opening in your webView	1094
第230.4节：注册令牌	1095	Section 230.4: Registration token	1095
第230.5节：订阅主题	1096	Section 230.5: Subscribe to a topic	1096
第231章：Firebase 实时数据库	1097	Chapter 231: Firebase Realtime DataBase	1097
第231.1节：快速设置	1097	Section 231.1: Quick setup	1097
第231.2节：Firebase 实时数据库事件处理器	1097	Section 231.2: Firebase Realtime DataBase event handler	1097
第231.3节：理解 Firebase JSON 数据库	1098	Section 231.3: Understanding firebase JSON database	1098
第231.4节：从 Firebase 获取数据	1099	Section 231.4: Retrieving data from firebase	1099
第231.5节：监听子节点更新	1100	Section 231.5: Listening for child updates	1100
第231.6节：使用分页检索数据	1101	Section 231.6: Retrieving data with pagination	1101
第231.7节：反规范化：扁平数据库结构	1102	Section 231.7: Denormalization: Flat Database Structure	1102
第231.8节：设计与理解如何从Firebase数据库检索实时数据	1104	Section 231.8: Designing and understanding how to retrieve realtime data from the Firebase Database	1104
第232章：Firebase应用索引	1107	Chapter 232: Firebase App Indexing	1107
第232.1节：支持Http URL	1107	Section 232.1: Supporting Http URLs	1107
第232.2节：添加AppIndexing API	1108	Section 232.2: Add AppIndexing API	1108
第233章：Firebase崩溃报告	1110	Chapter 233: Firebase Crash Reporting	1110
第233.1节：如何报告错误	1110	Section 233.1: How to report an error	1110

第233.2节：如何将Firebase崩溃报告添加到您的应用	1110	Section 233.2: How to add Firebase Crash Reporting to your app	1110
第234章：Twitter API	1112	Chapter 234: Twitter APIs	1112
第234.1节：创建带有回调的Twitter登录按钮	1112	Section 234.1: Creating login with twitter button and attach a callback to it	1112
第235章：YouTube-API	1114	Chapter 235: Youtube-API	1114
第235.1节：继承YouTubeBaseActivity的活动	1114	Section 235.1: Activity extending YouTubeBaseActivity	1114
第235.2节：在Android上使用YouTube数据API	1115	Section 235.2: Consuming YouTube Data API on Android	1115
第235.3节：启动StandAlonePlayerActivity	1117	Section 235.3: Launching StandAlonePlayerActivity	1117
第235.4节：竖屏活动中的YoutubePlayerFragment	1118	Section 235.4: YoutubePlayerFragment in portrait Activity	1118
第235.5节：YouTube播放器API	1120	Section 235.5: YouTube Player API	1120
第236章：集成谷歌登录	1123	Chapter 236: Integrate Google Sign In	1123
第236.1节：使用辅助类实现谷歌登录	1123	Section 236.1: Google Sign In with Helper class	1123
第237章：安卓上的谷歌登录集成	1126	Chapter 237: Google signin integration on android	1126
第237.1节：在项目中集成谷歌认证（获取配置文件）	1126	Section 237.1: Integration of google Auth in your project. (Get a configuration file)	1126
第237.2节：谷歌登录代码实现	1126	Section 237.2: Code Implementation Google SignIn	1126
第238章：谷歌感知API	1128	Chapter 238: Google Awareness APIs	1128
第238.1节：使用围栏API获取一定范围内的位置变化	1128	Section 238.1: Get changes for location within a certain range using Fence API	1128
第238.2节：使用快照API获取当前位置	1129	Section 238.2: Get current location using Snapshot API	1129
第238.3节：使用围栏API获取用户活动变化	1129	Section 238.3: Get changes in user activity with Fence API	1129
第238.4节：使用快照API获取当前用户活动	1130	Section 238.4: Get current user activity using Snapshot API	1130
第238.5节：使用快照API获取耳机状态	1130	Section 238.5: Get headphone state with Snapshot API	1130
第238.6节：使用快照API获取附近地点	1131	Section 238.6: Get nearby places using Snapshot API	1131
第238.7节：使用快照API获取当前天气	1131	Section 238.7: Get current weather using Snapshot API	1131
第239章：Android版Google地图API v2	1132	Chapter 239: Google Maps API v2 for Android	1132
第239.1节：自定义谷歌地图样式	1132	Section 239.1: Custom Google Map Styles	1132
第239.2节：默认谷歌地图活动	1143	Section 239.2: Default Google Map Activity	1143
第239.3节：在谷歌地图中显示当前位置	1144	Section 239.3: Show Current Location in a Google Map	1144
第239.4节：更改偏移量	1150	Section 239.4: Change Offset	1150
第239.5节：MapView：在现有布局中嵌入谷歌地图	1150	Section 239.5: MapView: embedding a GoogleMap in an existing layout	1150
第239.6节：获取调试SHA1指纹	1152	Section 239.6: Get debug SHA1 fingerprint	1152
第239.7节：向地图添加标记	1153	Section 239.7: Adding markers to a map	1153
第239.8节：UI设置	1153	Section 239.8: UISettings	1153
第239.9节：信息窗口点击监听器	1154	Section 239.9: InfoWindow Click Listener	1154
第239.10节：获取证书密钥库文件的SH1指纹	1155	Section 239.10: Obtaining the SH1-Fingerprint of your certificate keystore file	1155
第239.11节：点击地图时不启动谷歌地图（简化模式）	1156	Section 239.11: Do not launch Google Maps when the map is clicked (lite mode)	1156
第240章：谷歌云端硬盘API	1157	Chapter 240: Google Drive API	1157
第240.1节：在安卓中集成谷歌云端硬盘	1157	Section 240.1: Integrate Google Drive in Android	1157
第240.2节：在谷歌云端硬盘上创建文件	1165	Section 240.2: Create a File on Google Drive	1165
第241章：展示谷歌广告	1168	Chapter 241: Displaying Google Ads	1168
第241.1节：添加插页广告	1168	Section 241.1: Adding Interstitial Ad	1168
第241.2节：基础广告设置	1169	Section 241.2: Basic Ad Setup	1169
第242章：AdMob	1171	Chapter 242: AdMob	1171
第242.1节：实施	1171	Section 242.1: Implementing	1171
第243章：Google Play 商店	1173	Chapter 243: Google Play Store	1173
第243.1节：打开您的应用的 Google Play 商店列表	1173	Section 243.1: Open Google Play Store Listing for your app	1173
第243.2节：打开包含您发布者账户所有应用的 Google Play 商店列表	1173	Section 243.2: Open Google Play Store with the list of all applications from your publisher account	1173
第244章：为发布签署您的安卓应用	1175	Chapter 244: Sign your Android App for Release	1175
第244.1节：签署您的应用	1175	Section 244.1: Sign your App	1175
第244.2节：配置build.gradle中的签名配置	1176	Section 244.2: Configure the build.gradle with signing configuration	1176

第245章：TensorFlow	1178	Chapter 245: TensorFlow	1178
第245.1节：如何使用	1178	Section 245.1: How to use	1178
第246章：Android Vk SDK	1179	Chapter 246: Android Vk Sdk	1179
第246.1节：初始化和登录	1179	Section 246.1: Initialization and login	1179
第247章：项目SDK版本	1181	Chapter 247: Project SDK versions	1181
第247.1节：定义项目SDK版本	1181	Section 247.1: Defining project SDK versions	1181
第248章：Android版Facebook SDK	1182	Chapter 248: Facebook SDK for Android	1182
第248.1节：如何在Android中添加Facebook登录	1182	Section 248.1: How to add Facebook Login in Android	1182
第248.2节：为Facebook登录创建自定义按钮	1184	Section 248.2: Create your own custom button for Facebook login	1184
第248.3节：Facebook登录/注册实现的简约指南	1185	Section 248.3: A minimalistic guide to Facebook login/signup implementation	1185
第248.4节：设置权限以访问Facebook个人资料中的数据	1186	Section 248.4: Setting permissions to access data from the Facebook profile	1186
第248.5节：退出Facebook登录	1186	Section 248.5: Logging out of Facebook	1186
第249章：线程	1187	Chapter 249: Thread	1187
第249.1节：线程示例及其描述	1187	Section 249.1: Thread Example with its description	1187
第249.2节：从后台线程更新用户界面	1187	Section 249.2: Updating the UI from a Background Thread	1187
第250章：异步任务（AsyncTask）	1189	Chapter 250: AsyncTask	1189
第250.1节：基本用法	1189	Section 250.1: Basic Usage	1189
第250.2节：将Activity作为弱引用传递以避免内存泄漏	1191	Section 250.2: Pass Activity as WeakReference to avoid memory leaks	1191
第250.3节：使用AsyncTask在Android中下载图片	1192	Section 250.3: Download Image using AsyncTask in Android	1192
第250.4节：取消AsyncTask	1195	Section 250.4: Canceling AsyncTask	1195
第250.5节：AsyncTask：任务的串行执行和并行执行	1195	Section 250.5: AsyncTask: Serial Execution and Parallel Execution of Task	1195
第250.6节：执行顺序	1198	Section 250.6: Order of execution	1198
第250.7节：发布进度	1198	Section 250.7: Publishing progress	1198
第251章：使用Espresso测试UI	1200	Chapter 251: Testing UI with Espresso	1200
第251.1节：Espresso概述	1200	Section 251.1: Overall Espresso	1200
第251.2节：Espresso简单UI测试	1202	Section 251.2: Espresso simple UI test	1202
第251.3节：打开关闭DrawerLayout	1205	Section 251.3: Open Close DrawerLayout	1205
第251.4节：设置Espresso	1206	Section 251.4: Set Up Espresso	1206
第251.5节：对视图执行操作	1207	Section 251.5: Performing an action on a view	1207
第251.6节：使用onView查找视图	1207	Section 251.6: Finding a view with onView	1207
第251.7节：创建Espresso测试类	1207	Section 251.7: Create Espresso Test Class	1207
第251.8节：向上导航	1208	Section 251.8: Up Navigation	1208
第251.9节：将一组测试类归为一个测试套件	1208	Section 251.9: Group a collection of test classes in a test suite	1208
第251.10节：Espresso自定义匹配器	1209	Section 251.10: Espresso custom matchers	1209
第252章：编写UI测试 - 安卓	1212	Chapter 252: Writing UI tests - Android	1212
第252.1节：MockWebServer示例	1212	Section 252.1: MockWebServer example	1212
第252.2节：IdlingResource	1214	Section 252.2: IdlingResource	1214
第253章：使用JUnit进行安卓单元测试	1218	Chapter 253: Unit testing in Android with JUnit	1218
第253.1节：将业务逻辑移出Android组件	1218	Section 253.1: Moving Business Logic Out of Android Componenets	1218
第253.2节：创建本地单元测试	1220	Section 253.2: Creating Local unit tests	1220
第253.3节：JUnit入门	1221	Section 253.3: Getting started with JUnit	1221
第253.4节：异常	1224	Section 253.4: Exceptions	1224
第253.5节：静态导入	1225	Section 253.5: Static import	1225
第254章：使用UIAutomator进行应用间UI测试	1226	Chapter 254: Inter-app UI testing with UIAutomator	1226
第254.1节：准备你的项目并编写第一个UIAutomator测试	1226	Section 254.1: Prepare your project and write the first UIAutomator test	1226
第254.2节：使用UIAutomatorViewer编写更复杂的测试	1226	Section 254.2: Writing more complex tests using the UIAutomatorViewer	1226
第254.3节：创建UIAutomator测试套件	1228	Section 254.3: Creating a test suite of UIAutomator tests	1228
第255章：Lint警告	1229	Chapter 255: Lint Warnings	1229
第255.1节：在xml文件中使用tools:ignore	1229	Section 255.1: Using tools:ignore in xml files	1229

第255.2节：使用gradle配置LintOptions	1229	Section 255.2: Configure LintOptions with gradle	1229
第255.3节：配置Java和XML源文件中的lint检查	1230	Section 255.3: Configuring lint checking in Java and XML source files	1230
第255.4节：如何配置lint.xml文件	1230	Section 255.4: How to configure the lint.xml file	1230
第255.5节：标记抑制警告	1231	Section 255.5: Mark Suppress Warnings	1231
第255.6节：导入资源时无“已弃用”错误	1231	Section 255.6: Importing resources without "Deprecated" error	1231
第256章：性能优化	1233	Chapter 256: Performance Optimization	1233
第256.1节：使用ViewHolder模式缓存视图查找	1233	Section 256.1: Save View lookups with the ViewHolder pattern	1233
第257章：Android内核优化	1234	Chapter 257: Android Kernel Optimization	1234
第257.1节：低内存配置	1234	Section 257.1: Low RAM Configuration	1234
第257.2节：如何添加CPU调节器	1234	Section 257.2: How to add a CPU Governor	1234
第257.3节：输入输出调度器	1236	Section 257.3: I/O Schedulers	1236
第258章：内存泄漏	1237	Chapter 258: Memory Leaks	1237
第258.1节：避免使用AsyncTask导致的Activity泄漏	1237	Section 258.1: Avoid leaking Activities with AsyncTask	1237
第258.2节：常见内存泄漏及其修复方法	1238	Section 258.2: Common memory leaks and how to fix them	1238
第258.3节：使用LeakCanary库检测内存泄漏	1239	Section 258.3: Detect memory leaks with the LeakCanary library	1239
第258.4节：Activity中的匿名回调	1239	Section 258.4: Anonymous callback in activities	1239
第258.5节：静态类中的活动上下文	1240	Section 258.5: Activity Context in static classes	1240
第258.6节：避免带有监听器的活动泄漏	1241	Section 258.6: Avoid leaking Activities with Listeners	1241
第258.7节：避免匿名类、Handler、Timer任务、线程的内存泄漏	1246	Section 258.7: Avoid memory leaks with Anonymous Class, Handler, Timer Task, Thread	1246
第259章：使用图标字体提升安卓性能	1248	Chapter 259: Enhancing Android Performance Using Icon Fonts	1248
第259.1节：如何集成图标字体	1248	Section 259.1: How to integrate Icon fonts	1248
第259.2节：带图标字体的TabLayout	1250	Section 259.2: TabLayout with icon fonts	1250
第260章：位图缓存	1252	Chapter 260: Bitmap Cache	1252
第260.1节：使用LRU缓存的位图缓存	1252	Section 260.1: Bitmap Cache Using LRU Cache	1252
第261章：有效加载位图	1253	Chapter 261: Loading Bitmaps Effectively	1253
第261.1节：使用Intent从Android设备资源加载图像	1253	Section 261.1: Load the Image from Resource from Android Device. Using Intents	1253
第262章：异常	1258	Chapter 262: Exceptions	1258
第262.1节：ActivityNotFoundException	1258	Section 262.1: ActivityNotFoundException	1258
第262.2节：内存溢出错误	1258	Section 262.2: OutOfMemoryError	1258
第262.3节：为意外异常注册自定义处理程序	1258	Section 262.3: Registering own Handler for unexpected exceptions	1258
第262.4节：未捕获异常（UncaughtException）	1260	Section 262.4: UncaughtException	1260
第262.5节：主线程网络异常（NetworkOnMainThreadException）	1260	Section 262.5: NetworkOnMainThreadException	1260
第262.6节：Dex异常（DexException）	1262	Section 262.6: DexException	1262
第263章：日志记录与使用Logcat	1263	Chapter 263: Logging and using Logcat	1263
第263.1节：过滤Logcat输出	1263	Section 263.1: Filtering the logcat output	1263
第263.2节：日志记录	1264	Section 263.2: Logging	1264
第263.3节：使用Logcat	1266	Section 263.3: Using the Logcat	1266
第263.4节：从Logcat直接链接到源代码的日志	1267	Section 263.4: Log with link to source directly from Logcat	1267
第263.5节：清除日志	1267	Section 263.5: Clear logs	1267
第263.6节：Android Studio的使用	1267	Section 263.6: Android Studio usage	1267
第263.7节：生成日志代码	1268	Section 263.7: Generating Logging code	1268
第264章：ADB (Android调试桥)	1270	Chapter 264: ADB (Android Debug Bridge)	1270
第264.1节：通过WiFi将ADB连接到设备	1270	Section 264.1: Connect ADB to a device via WiFi	1270
第264.2节：在多设备环境中向特定设备发送ADB命令	1272	Section 264.2: Direct ADB command to specific device in a multi-device setting	1272
第264.3节：从设备显示屏截取屏幕截图和视频（仅限KitKat）	1272	Section 264.3: Taking a screenshot and video (for kitkat only) from a device display	1272
第264.4节：从设备拉取（推送）文件	1273	Section 264.4: Pull (push) files from (to) the device	1273
第264.5节：打印已连接设备的详细列表	1274	Section 264.5: Print verbose list of connected devices	1274
第264.6节：查看logcat日志	1274	Section 264.6: View logcat	1274
第264.7节：查看并拉取应用的缓存文件	1275	Section 264.7: View and pull cache files of an app	1275

第264.8节：清除应用数据	1275
第264.9节：查看设备上应用的内部数据 (data/data/<sample.package.id>)	1276
第264.10节：安装并运行应用程序	1276
第264.11节：发送广播	1276
第264.12节：备份	1277
第264.13节：查看可用设备	1278
第264.14节：通过IP连接设备	1278
第264.15节：在Linux系统上安装ADB	1279
第264.16节：查看活动栈	1279
第264.17节：重启设备	1279
第264.18节：读取设备信息	1280
第264.19节：列出Android 6.0上所有需要用户运行时授权的权限	1280
第264.20节：打开/关闭Wifi	1280
第264.21节：启动/停止adb	1280
第265章：Android中使用资源的本地化	1281
第265.1节：“res”目录下各文件夹的配置类型和限定符名称	1281
第265.2节：为您的Android应用添加翻译	1282
第265.3节：“res”文件夹下的资源目录类型	1284
第265.4节：通过编程方式更改Android应用的语言环境	1284
第265.5节：货币	1287
第266章：将越南语字符串转换为英文字符串 Android	1288
第266.1节：示例：	1288
第266.2节：将越南语字符串转换为无音调字符串	1288
鸣谢	1289
你可能也喜欢	1301

Section 264.8: Clear application data	1275
Section 264.9: View an app's internal data (data/data/<sample.package.id>) on a device	1276
Section 264.10: Install and run an application	1276
Section 264.11: Sending broadcast	1276
Section 264.12: Backup	1277
Section 264.13: View available devices	1278
Section 264.14: Connect device by IP	1278
Section 264.15: Install ADB on Linux system	1279
Section 264.16: View activity stack	1279
Section 264.17: Reboot device	1279
Section 264.18: Read device information	1280
Section 264.19: List all permissions that require runtime grant from users on Android 6.0	1280
Section 264.20: Turn on/off Wifi	1280
Section 264.21: Start/stop adb	1280
Chapter 265: Localization with resources in Android	1281
Section 265.1: Configuration types and qualifier names for each folder under the "res" directory	1281
Section 265.2: Adding translation to your Android app	1282
Section 265.3: Type of resource directories under the "res" folder	1284
Section 265.4: Change locale of android application programmatically	1284
Section 265.5: Currency	1287
Chapter 266: Convert vietnamese string to english string Android	1288
Section 266.1: example:	1288
Section 266.2: Chuyển chuỗi Tiếng Việt thành chuỗi không dấu	1288
Credits	1289
You may also like	1301

About

欢迎随意免费分享此PDF，
本书最新版本可从以下网址下载：

<https://goalkicker.com/AndroidBook>

本Android™ 专业人士笔记一书汇编自[Stack Overflow Documentation](#)，内容由Stack Overflow的优秀贡献者撰写。
文本内容采用知识共享署名-相同方式共享许可协议发布，详见本书末尾对各章节贡献者的致谢。图片版权归各自所有者所有，除非另有说明。

本书为非官方免费教材，旨在教育用途，与官方Android™组织或公司及Stack Overflow无关。所有商标及注册商标均为其各自公司所有。

本书所提供信息不保证正确或准确，使用风险自负。

请将反馈和更正发送至web@petercv.com

Please feel free to share this PDF with anyone for free,
latest version of this book can be downloaded from:

<https://goalkicker.com/AndroidBook>

This *Android™ Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow.
Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official Android™ group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to web@petercv.com

第1章：Android入门

版本	API 级别	版本代码	发布日期
1.0	1	基础版	2008-09-23
1.1	2	基础版_1.1	2009-02-09
1.5	3	杯形蛋糕	2009-04-27
1.6	4	甜甜圈	2009-09-15
2.0	5	闪电泡芙	2009-10-26
2.0.1	6	闪电泡芙_0.1	2009-12-03
2.1.x	7	闪电泡芙_MR1	2010-01-12
2.2.x	8	FROYO	2010-05-20
2.3	9	GINGERBREAD	2010-12-06
2.3.3	10	GINGERBREAD_MR1	2011-02-09
3.0.x	11	蜂巢	2011-02-22
3.1.x	12	蜂巢_MR1	2011-05-10
3.2.x	13	蜂巢_MR2	2011-07-15
4.0	14	冰淇淋三明治	2011-10-18
4.0.3	15	ICE_CREAM SANDWICH_MR1	2011-12-16
4.1	16	果冻豆	2012-07-09
4.2	17	果冻豆_MR1	2012-11-13
4.3	18	果冻豆_MR2	2013-07-24
4.4	19	奇巧	2013-10-31
4.4W	20	KITKAT_WATCH	2014-06-25
5.0	21	LOLLIPOP	2014-11-12
5.1	22	LOLLIPOP_MR1	2015-03-09
6.0	23	M_(棉花糖)	2015-10-05
7.0	24	N_(牛轧糖)	2016-08-22
7.1	25	N_MR1_(牛轧糖_MR1)	2016-10-04
8.0	26	O_(开发者预览版_4)	2017-07-24

第1.1节：创建新项目

设置 Android Studio

首先设置 Android Studio，然后打开它。现在，你已经准备好制作你的第一个 Android 应用程序了！

注意：本指南基于 Android Studio 2.2，但其他版本的流程基本相同。

配置您的项目

基本配置

您可以通过两种方式开始一个新项目：

- 点击欢迎界面上的开始一个新的Android Studio项目。
- 如果您已经打开了一个项目，请导航到文件→新建项目。

接下来，您需要通过填写一些字段来描述您的应用程序：

Chapter 1: Getting started with Android

Version	API Level	Version Code	Release Date
1.0	1	BASE	2008-09-23
1.1	2	BASE_1_1	2009-02-09
1.5	3	CUPCAKE	2009-04-27
1.6	4	DONUT	2009-09-15
2.0	5	ECLAIR	2009-10-26
2.0.1	6	ECLAIR_0_1	2009-12-03
2.1.x	7	ECLAIR_MR1	2010-01-12
2.2.x	8	FROYO	2010-05-20
2.3	9	GINGERBREAD	2010-12-06
2.3.3	10	GINGERBREAD_MR1	2011-02-09
3.0.x	11	HONEYCOMB	2011-02-22
3.1.x	12	HONEYCOMB_MR1	2011-05-10
3.2.x	13	HONEYCOMB_MR2	2011-07-15
4.0	14	ICE_CREAM SANDWICH	2011-10-18
4.0.3	15	ICE_CREAM SANDWICH_MR1	2011-12-16
4.1	16	JELLY_BEAN	2012-07-09
4.2	17	JELLY_BEAN_MR1	2012-11-13
4.3	18	JELLY_BEAN_MR2	2013-07-24
4.4	19	KITKAT	2013-10-31
4.4W	20	KITKAT_WATCH	2014-06-25
5.0	21	LOLLIPOP	2014-11-12
5.1	22	LOLLIPOP_MR1	2015-03-09
6.0	23	M_(Marshmallow)	2015-10-05
7.0	24	N_(Nougat)	2016-08-22
7.1	25	N_MR1_(Nougat_MR1)	2016-10-04
8.0	26	O_(Developer Preview_4)	2017-07-24

Section 1.1: Creating a New Project

Set up Android Studio

Start by setting up Android Studio and then open it. Now, you're ready to make your first Android App!

Note: this guide is based on Android Studio 2.2, but the process on other versions is mainly the same.

Configure Your Project

Basic Configuration

You can start a new project in two ways:

- Click Start a New Android Studio Project from the welcome screen.
- Navigate to File → New Project if you already have a project open.

Next, you need to describe your application by filling out some fields:

1. 应用名称 - 该名称将显示给用户。

示例：Hello World。您可以随时在AndroidManifest.xml文件中更改它。

2. 公司域名 - 这是您项目包名的限定符。

示例：stackoverflow.com.

3. 包名（又称applicationId） - 这是完全限定的项目包名。

它应遵循反向域名表示法（又称反向DNS）：顶级域名.公司域名。

[公司部分.]应用名称。

示例：com.stackoverflow.android.helloworld或com.stackoverflow.helloworld。你可以随时通过在gradle文件中重写来更改你的applicationId。

除非你不打算将应用提交到
Google Play商店，否则不要使用默认前缀“com.example”。包名将在Google Play中作为你的唯一applicationId。

4. 项目位置- 这是存储你的项目的目录。

1. Application Name - This name will be shown to the user.

Example: Hello World. You can always change it later in AndroidManifest.xml file.

2. Company Domain - This is the qualifier for your project's package name.

Example: stackoverflow.com.

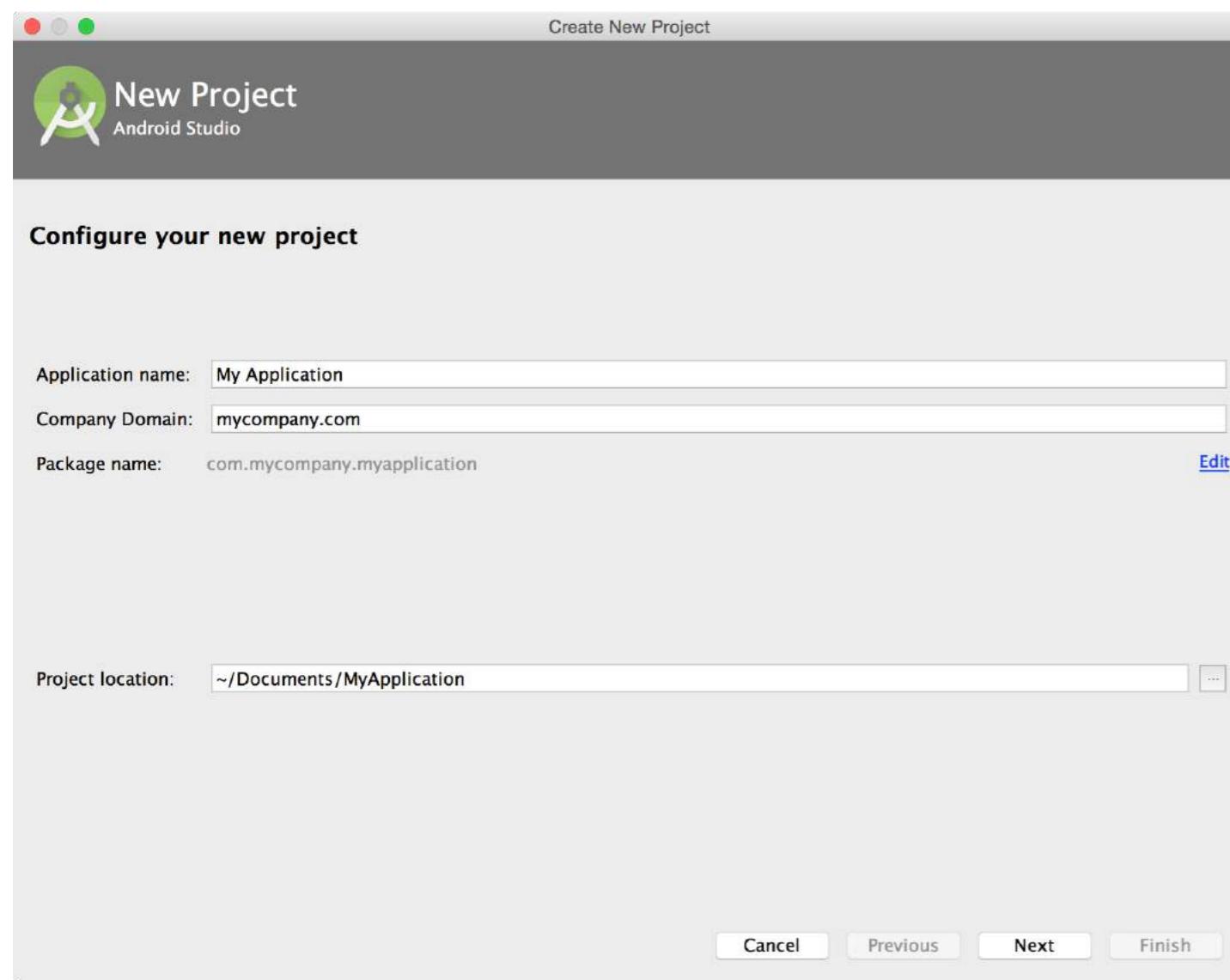
3. Package Name (aka applicationId) - This is the *fully qualified* project package name.

It should follow *Reverse Domain Name Notation* (aka *Reverse DNS*): *Top Level Domain . Company Domain . [Company Segment .] Application Name*.

Example: com.stackoverflow.android.helloworld or com.stackoverflow.helloworld. You can always change your applicationId by overriding it in your gradle file.

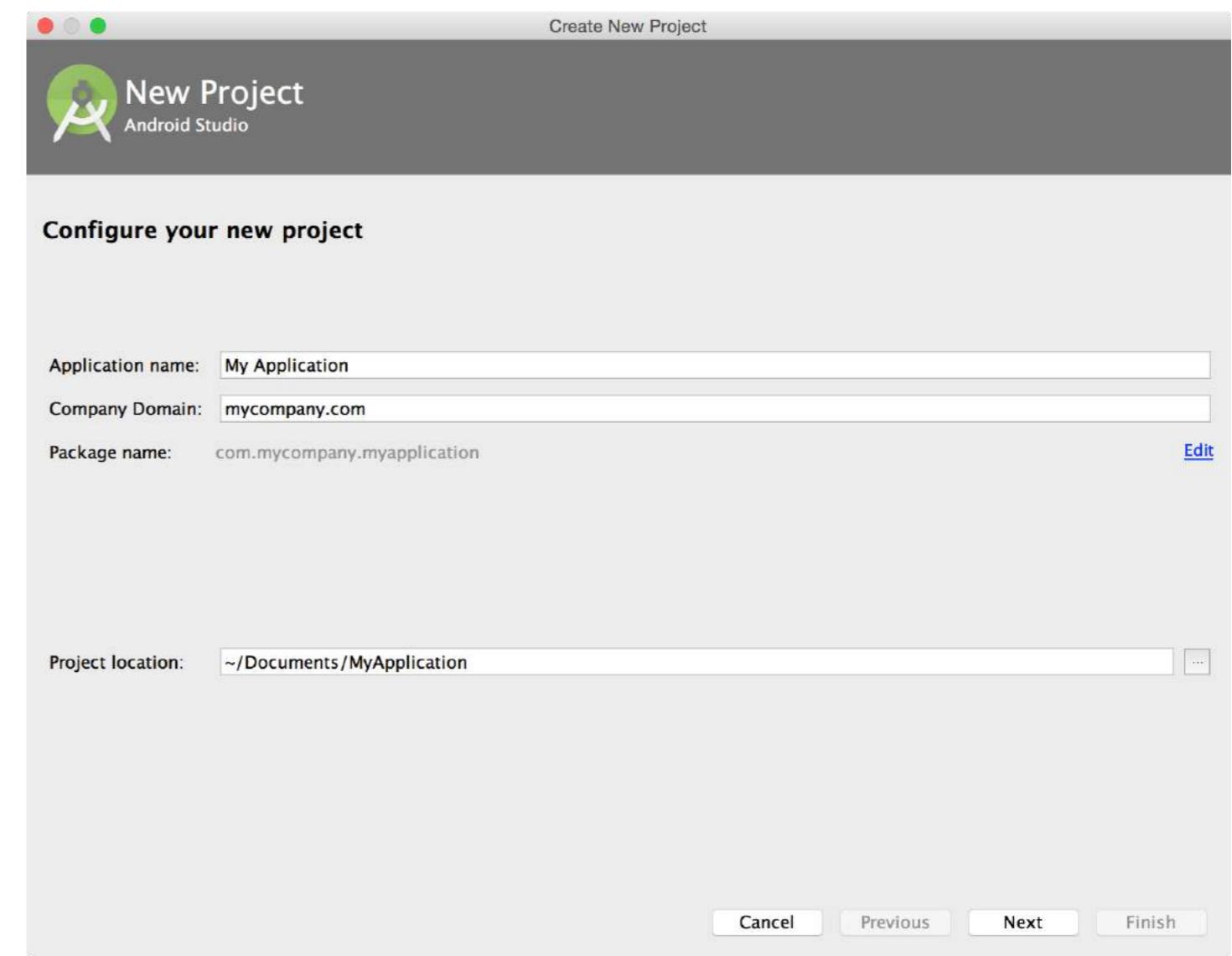
Don't use the default prefix "com.example" unless you don't intend to submit your application to the Google Play Store. The package name will be your unique **applicationId** in Google Play.

4. Project Location - This is the directory where your project will be stored.



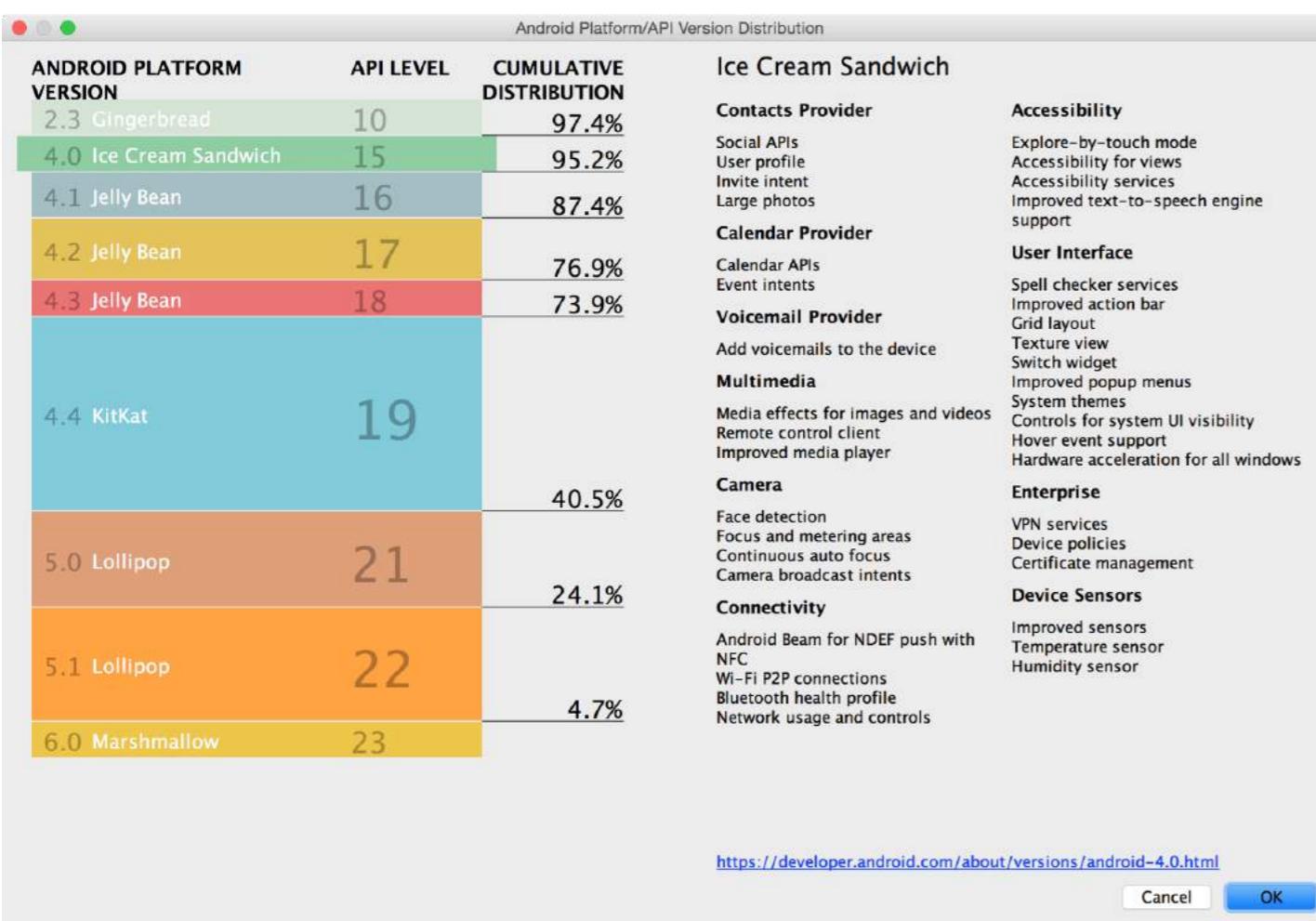
选择设备类型和API级别

下一个窗口让你选择应用支持的设备类型，如手机、平板、电视、Wear和Google Glass。所选设备类型将成为项目中的应用模块。对于每种设备类型，你还可以选择该应用的API级别。要获取更多信息，请点击[帮助我选择](#)



Select Form Factors and API Level

The next window lets you select the form factors supported by your app, such as phone, tablet, TV, Wear, and Google Glass. The selected form factors become the app modules within the project. For each form factor, you can also select the API Level for that app. To get more information, click **Help me choose**



点击“帮助我选择”时显示的当前安卓版本分布图表。

安卓平台分布窗口显示运行各版本安卓的移动设备分布，如图2所示。点击某个API级别，可以查看对应安卓版本引入的功能列表。这有助于你选择包含应用所需所有功能的最低API级别，从而覆盖尽可能多的设备。然后点击确定。

现在，选择应用将支持的平台和安卓SDK版本。

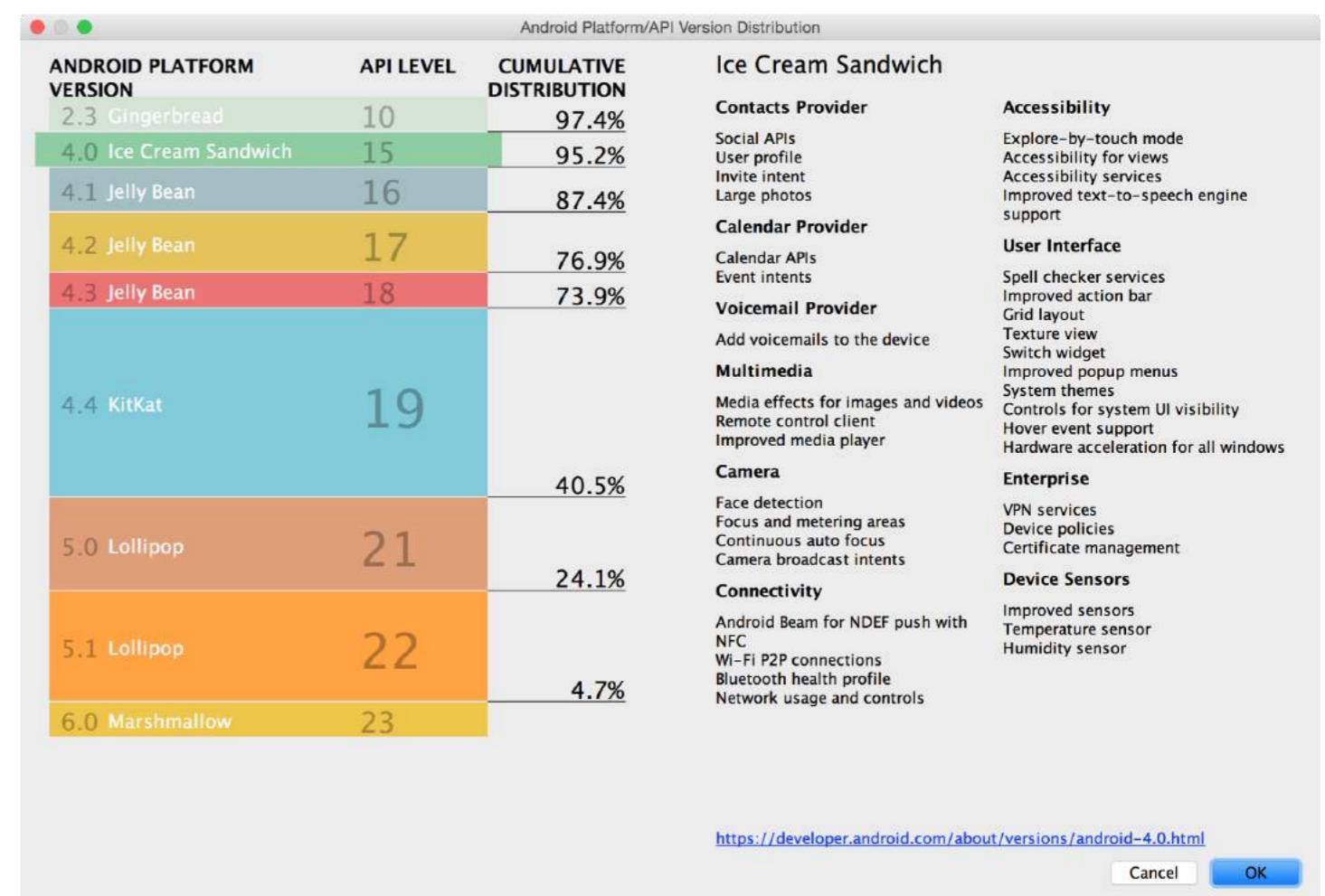
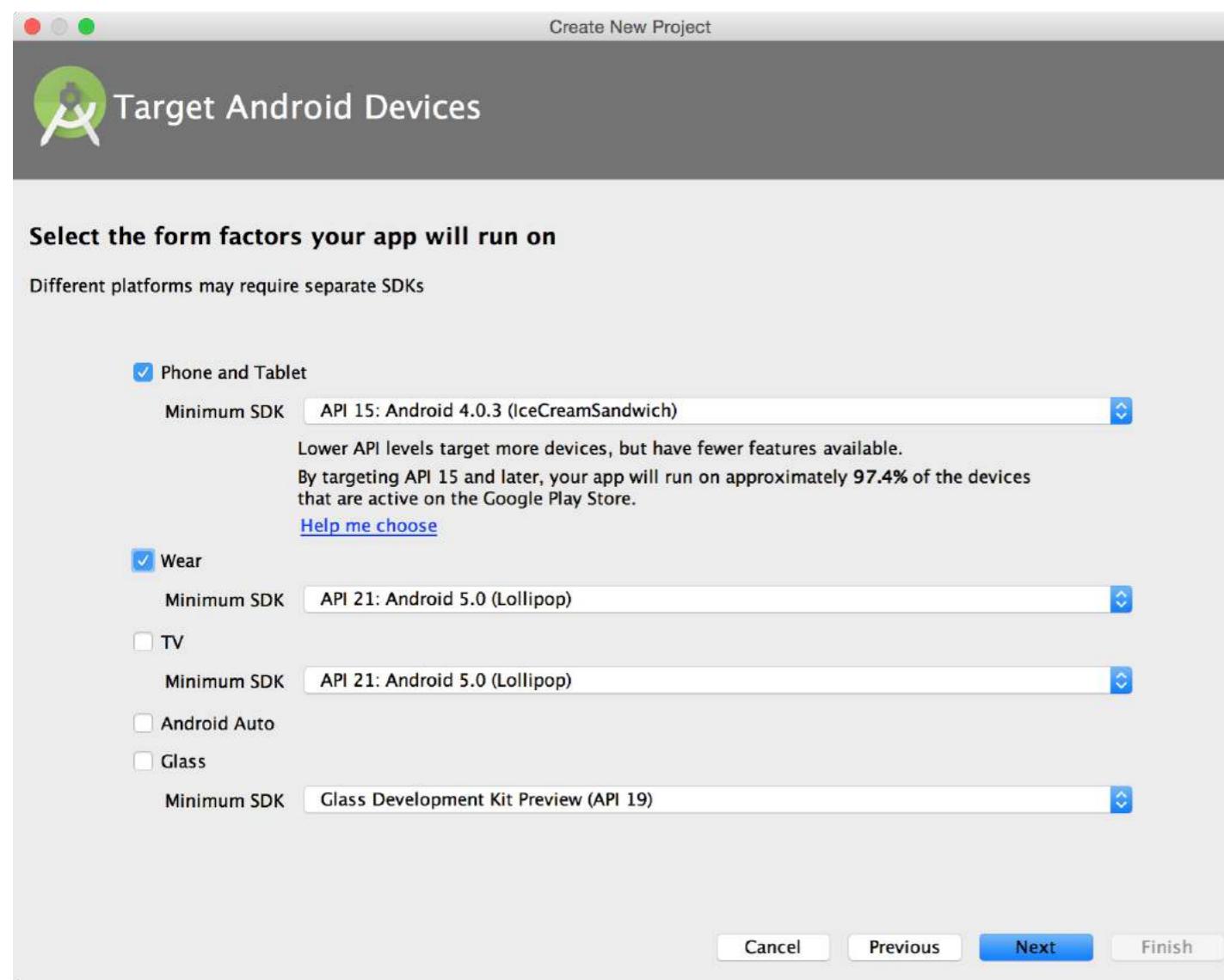


Chart of the current Android version distributions, shown when you click Help me choose.

The Android Platform Distribution window shows the distribution of mobile devices running each version of Android, as shown in Figure 2. Click on an API level to see a list of features introduced in the corresponding version of Android. This helps you choose the minimum API Level that has all the features that your apps needs, so you can reach as many devices as possible. Then click **OK**.

Now, choose what platforms and version of Android SDK the application will support.



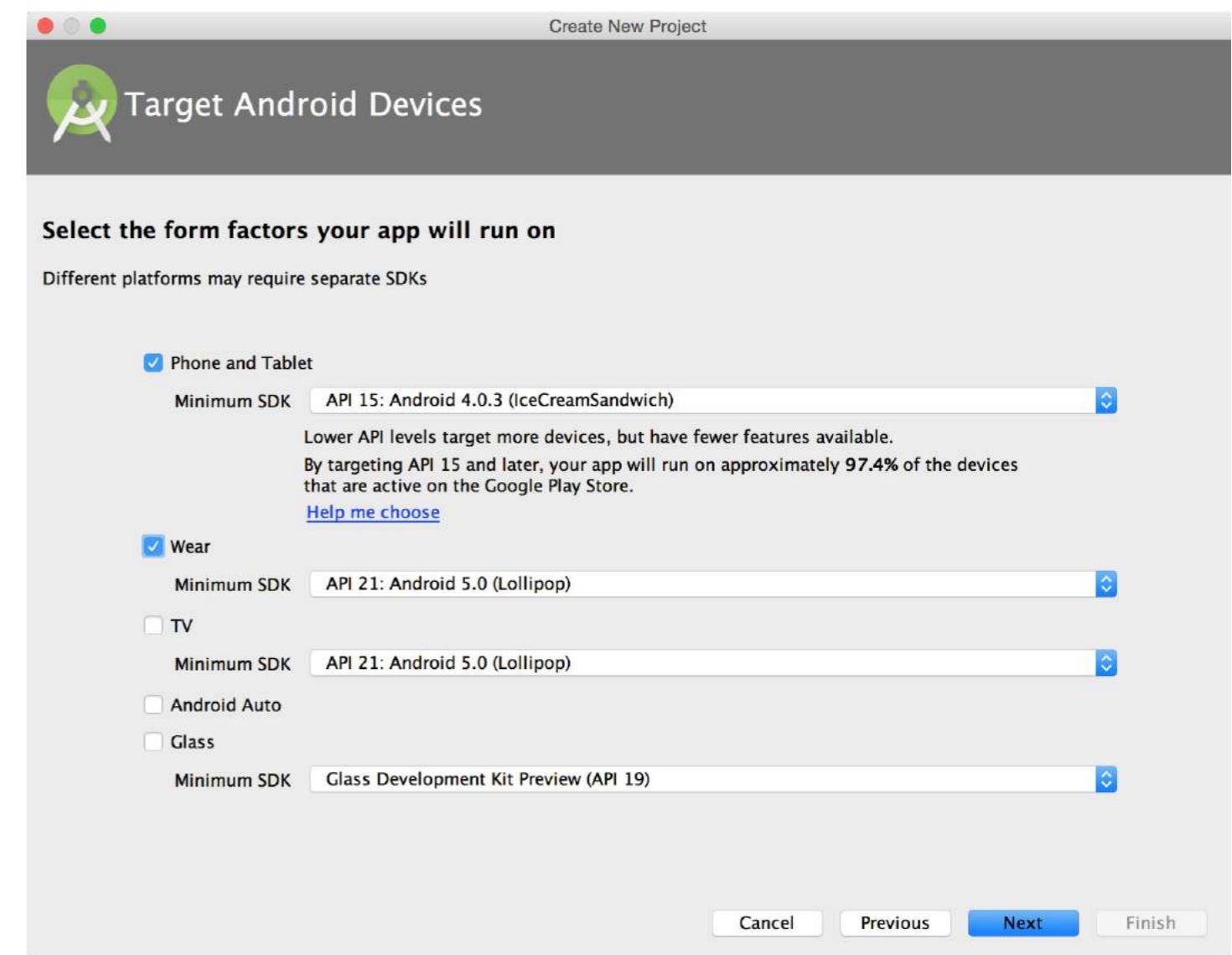
目前，仅选择手机和平板。

最低SDK是你的应用的下限。它是Google Play商店用来确定应用可安装设备的信号之一。例如，Stack Exchange的应用支持安卓4.1及以上版本。

ADDITIONAL INFORMATION

Updated September 28, 2016	Installs 100,000 - 500,000	Current Version 1.0.89
Requires Android 4.1 and up	Content Rating Rated for 12+ Parental Guidance Recommended Learn more	Interactive Elements Users Interact
Permissions View details	Report Flag as inappropriate	Offered By Stack Exchange

安卓工作室会告诉你（大致）在指定最低SDK的情况下，支持的设备百分比。



For now, select only *Phone and Tablet*.

The **Minimum SDK** is the lower bound for your app. It is one of the signals the Google Play Store uses to determine which devices an app can be installed on. For example, [Stack Exchange's app](#) supports Android 4.1+.

ADDITIONAL INFORMATION

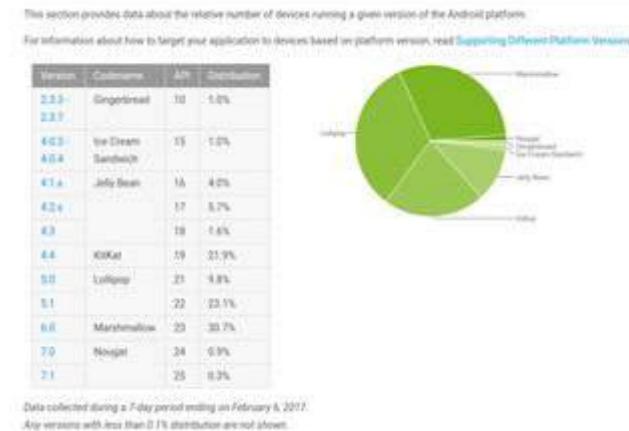
Updated September 28, 2016	Installs 100,000 - 500,000	Current Version 1.0.89
Requires Android 4.1 and up	Content Rating Rated for 12+ Parental Guidance Recommended Learn more	Interactive Elements Users Interact
Permissions View details	Report Flag as inappropriate	Offered By Stack Exchange

Android Studio will tell you (approximately) what percentage of devices will be supported given the specified minimum SDK.

较低的API级别覆盖更多设备，但可用功能较少。

在决定最低SDK时，应考虑仪表盘统计数据，该数据提供了过去一周全球访问Google Play商店设备的版本信息。

Platform Versions



来源：安卓开发者网站上的仪表盘。

添加一个活动

现在我们将为我们的应用选择一个默认活动。在安卓中，Activity 是呈现给用户的单个界面。一个应用可以包含多个活动并在它们之间导航。对于本例，选择空白活动然后点击下一步。

这里，如果你愿意，可以更改活动的名称和布局。一个好的做法是将Activity作为活动名称的后缀，将activity_作为布局名称的前缀。如果我们保持默认，Android Studio会为我们生成一个名为MainActivity的活动，以及一个名为activity_main的布局文件。现在点击Finish。

Android Studio将创建并配置我们的项目，这可能会根据系统情况花费一些时间。

检查项目

为了理解Android的工作原理，让我们看看为我们创建的一些文件。

在Android Studio的左侧窗格中，我们可以看到我们的Android应用程序的结构。

Lower API levels target more devices but have fewer features available.

When deciding on the **Minimum SDK**, you should consider the [Dashboards stats](#), which will give you version information about the devices that visited the Google Play Store globally in the last week.

Platform Versions



From: [Dashboards](#) on Android Developer website.

Add an activity

Now we are going to select a default activity for our application. In Android, an **Activity** is a single screen that will be presented to the user. An application can house multiple activities and navigate between them. For this example, choose **Empty Activity** and click next.

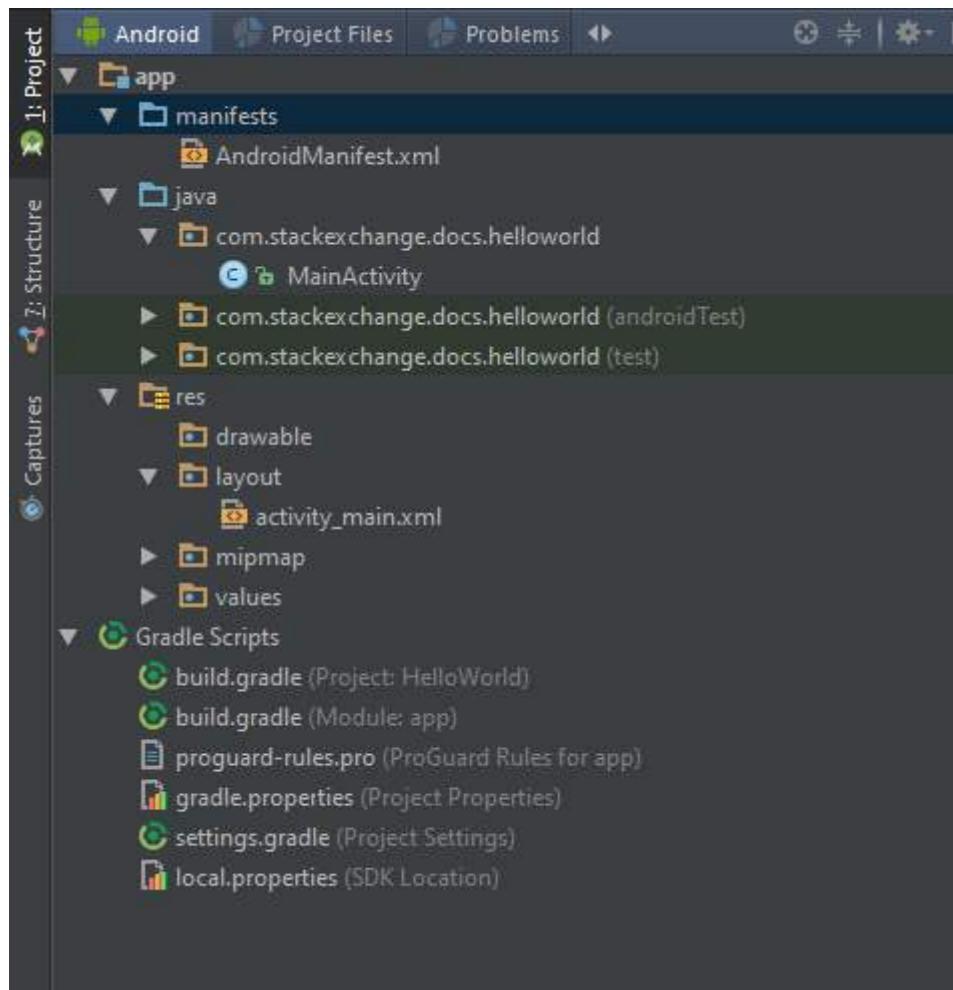
Here, if you wish, you can change the name of the activity and layout. A good practice is to keep Activity as a suffix for the activity name, and activity_ as a prefix for the layout name. If we leave these as the default, Android Studio will generate an activity for us called **MainActivity**, and a layout file called **activity_main**. Now click **Finish**.

Android Studio will create and configure our project, which can take some time depending on the system.

Inspecting the Project

To understand how Android works, let's take a look at some of the files that were created for us.

On the left pane of Android Studio, we can see the [structure of our Android application](#).

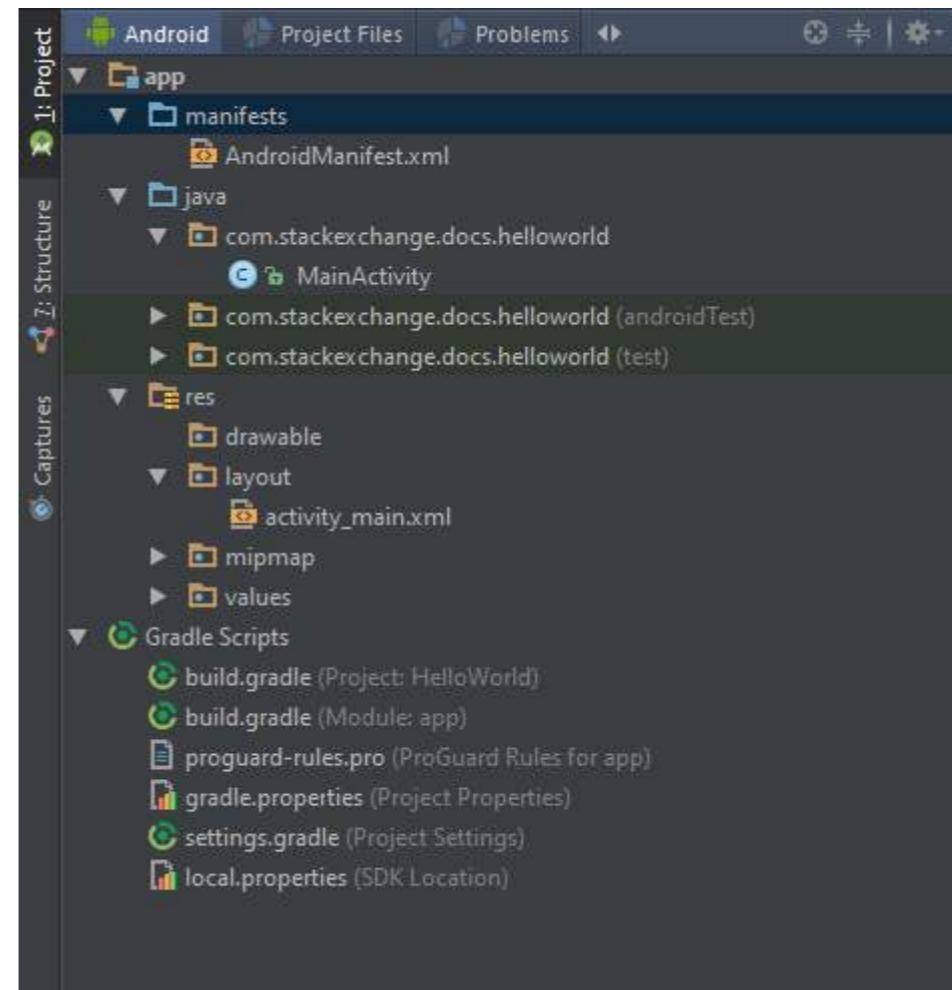


首先，双击打开AndroidManifest.xml。Android清单文件描述了Android应用程序的一些基本信息。它包含了我们活动的声明，以及一些更高级的组件。

如果应用程序需要访问受权限保护的功能，必须在清单中使用`<uses-permission>`元素声明所需权限。然后，当应用程序安装到设备上时，安装程序会通过检查签署应用程序证书的权限机构，并在某些情况下询问用户，来决定是否授予请求的权限。应用程序还可以使用权限保护自己的组件（活动、服务、广播接收器和内容提供者）。它可以使用Android定义的任何权限（列在`android.Manifest.permission`中）或其他应用声明的权限，或者定义自己的权限。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



First, let's open `AndroidManifest.xml` by double clicking it. The Android manifest file describes some of the basic information about an Android application. It contains the declaration of our activities, as well as some more advanced components.

If an application needs access to a feature protected by a permission, it must declare that it requires that permission with a `<uses-permission>` element in the manifest. Then, when the application is installed on the device, the installer determines whether or not to grant the requested permission by checking the authorities that signed the application's certificates and, in some cases, asking the user. An application can also protect its own components (activities, services, broadcast receivers, and content providers) with permissions. It can employ any of the permissions defined by Android (listed in `android.Manifest.permission`) or declared by other applications. Or it can define its own.

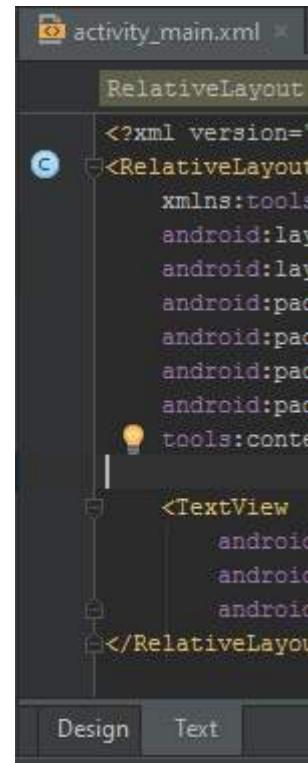
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
</application>  
</manifest>
```

接下来，让我们打开位于app/src/main/res/layout/目录下的activity_main.xml文件。该文件包含了MainActivity的视觉组件声明。你将看到视觉设计器，它允许你将元素拖放到选定的布局中。

你也可以通过点击Android Studio底部的“Text”切换到xml布局设计器，如下所示：



```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="com.stackexchange.docs.helloworld.MainActivity">  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello World!" />  
</RelativeLayout>
```

你会在这个布局中看到一个名为TextView的小部件，其android:text属性设置为"Hello World!"。这是一个文本块，当用户运行应用程序时会显示给用户。

你可以阅读更多关于布局和属性的内容。[Layouts and attributes](#)

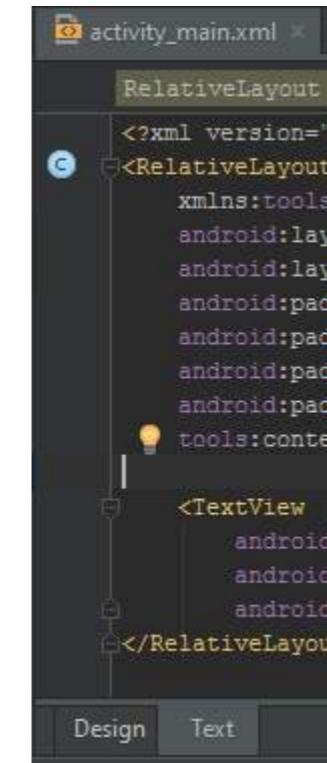
接下来，让我们看看MainActivity。这是为MainActivity生成的Java代码。

```
public class MainActivity extends AppCompatActivity {  
  
    // 当Activity启动时调用onCreate方法
```

```
</application>  
</manifest>
```

Next, let's open `activity_main.xml` which is located in `app/src/main/res/layout/`. This file contains declarations for the visual components of our MainActivity. You will see visual designer. This allows you to drag and drop elements onto the selected layout.

You can also switch to the xml layout designer by clicking "Text" at the bottom of Android Studio, as seen here:



```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="com.stackexchange.docs.helloworld.MainActivity">  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello World!" />  
</RelativeLayout>
```

You will see a widget called a TextView inside of this layout, with the android:text property set to "Hello World!". This is a block of text that will be shown to the user when they run the application.

You can read more about [Layouts and attributes](#).

Next, let's take a look at MainActivity. This is the Java code that has been generated for MainActivity.

```
public class MainActivity extends AppCompatActivity {  
  
    // The onCreate method is called when an Activity starts
```

```
// 我们将在这里设置布局
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // setContentView 设置 Activity 的布局为指定的 XML 布局
    // 在我们的例子中，我们使用的是 activity_main 布局
    setContentView(R.layout.activity_main);
}

}
```

正如我们在 Android 清单文件中定义的，MainActivity将在用户启动HelloWorld应用时默认启动。

最后，打开位于app/目录下名为build.gradle的文件。

Android Studio 使用构建系统Gradle来编译和构建 Android 应用和库。

```
apply plugin: 'com.android.application'

android {
    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'anotherPassword'
        }
    }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}
```

该文件包含有关构建和您的应用版本的信息，您还可以使用它来添加对外部库的依赖。目前，我们暂时不做任何更改。

建议始终选择可用的最新版本作为依赖项：

- [buildToolsVersion: 26.0.0](#)
- [com.android.support:appcompat-v7: 26.0.0 \(2017年7月\)](#)
- [firebase: 11.0.4 \(2017年8月\)](#)

```
// This is where we will set up our layout
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // setContentView sets the Activity's layout to a specified XML layout
    // In our case we are using the activity_main layout
    setContentView(R.layout.activity_main);
}

}
```

As defined in our Android manifest, MainActivity will launch by default when a user starts the HelloWorld app.

Lastly, open up the file named `build.gradle` located in `app/`.

Android Studio uses the build system **Gradle** to compile and build Android applications and libraries.

```
apply plugin: 'com.android.application'

android {
    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'anotherPassword'
        }
    }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}
```

This file contains information about the build and your app version, and you can also use it to add dependencies to external libraries. For now, let's not make any changes.

It is advisable to always select the latest version available for the dependencies:

- [buildToolsVersion: 26.0.0](#)
- [com.android.support:appcompat-v7: 26.0.0 \(July 2017\)](#)
- [firebase: 11.0.4 \(August 2017\)](#)

compileSdkVersion

compileSdkVersion 是告诉 Gradle 使用哪个版本的 Android SDK 来编译您的应用。使用新的 Android SDK 是使用该级别新增 API 的前提条件。

需要强调的是，更改您的 compileSdkVersion 不会改变运行时行为。虽然更改 compileSdkVersion 可能会出现新的编译器警告/错误，但 compileSdkVersion 不包含在您的 APK 中：它仅在编译时使用。

因此，强烈建议您始终使用最新的 SDK 进行编译。您将获得对现有代码的新编译检查的所有好处，避免使用新弃用的 API，并准备好使用新 API。

minSdkVersion

如果compileSdkVersion设置了可用的最新API，minSdkVersion则是应用的下限。minSdkVersion是Google Play商店用来确定应用可以安装在哪些用户设备上的信号之一。

它在开发过程中也起着重要作用：默认情况下，lint会针对你的项目运行，当你使用任何高于minSdkVersion的API时会发出警告，帮助你避免在运行时调用不存在的API的问题。在运行时检查系统版本是仅在较新平台版本上使用API的常见技术。

targetSdkVersion

targetSdkVersion是Android提供向前兼容性的主要方式，只有在更新了targetSdkVersion时才会应用行为更改。这允许你在处理行为更改之前使用新API。

更新以针对最新SDK应是每个应用的高优先级。这并不意味着你必须使用引入的每个新功能，也不应在未经测试的情况下盲目更新你的targetSdkVersion。

targetSDKVersion是Android版本，代表可用工具的上限。如果targetSDKVersion小于23，例如，应用即使在API 23+上运行，也不需要在运行时请求权限。targetSDKVersion不会阻止高于所选Android版本的Android版本运行该应用。

你可以找到关于Gradle插件的更多信息：

- 一个基本示例
- Android的Gradle插件及其包装器介绍
- build.gradle的配置及DSL方法介绍

运行应用程序

现在，让我们运行我们的HelloWorld应用程序。你可以运行一个Android虚拟设备（可以通过在Android Studio中使用AVD管理器设置，如下面的示例所述），或者通过USB线连接你自己的Android设备。

设置Android设备

要从Android Studio在你的Android设备上运行应用程序，必须在设备的设置中启用USB调试，位于开发者选项中。

设置>开发者选项>USB调试

如果设置中看不到开发者选项，进入关于手机并连续点击版本号七次

compileSdkVersion

compileSdkVersion is your way to tell *Gradle* what version of the Android SDK to compile your app with. Using the new Android SDK is a requirement to use any of the new APIs added in that level.

It should be emphasized that changing your compileSdkVersion does not change runtime behavior. While new compiler warnings/errors may be present when changing your compileSdkVersion, your compileSdkVersion is not included in your APK: it is purely used at compile time.

Therefore it is strongly recommended that you always compile with the latest SDK. You'll get all the benefits of new compilation checks on existing code, avoid newly deprecated APIs, and be ready to use new APIs.

minSdkVersion

If compileSdkVersion sets the newest APIs available to you, minSdkVersion is the *lower bound* for your app. The minSdkVersion is one of the signals the Google Play Store uses to determine which of a user's devices an app can be installed on.

It also plays an important role during development: by default lint runs against your project, warning you when you use any APIs above your minSdkVersion, helping you avoid the runtime issue of attempting to call an API that doesn't exist. Checking the system version at runtime is a common technique when using APIs only on newer platform versions.

targetSdkVersion

targetSdkVersion is the main way Android provides forward compatibility by not applying behavior changes unless the targetSdkVersion is updated. This allows you to use new APIs prior to working through the behavior changes. Updating to target the latest SDK should be a high priority for every app. That doesn't mean you have to use every new feature introduced nor should you blindly update your targetSdkVersion without testing.

targetSDKVersion is the version of Android which is the upper-limit for the available tools. If targetSDKVersion is less than 23, the app does not need to request permissions at runtime for an instance, even if the app is being run on API 23+. TargetSDKVersion does **not** prevent android versions above the picked Android version from running the app.

You can find more info about the Gradle plugin:

- A basic example
- Introduction to the Gradle plugin for android and the wrapper
- Introduction to the configuration of the build.gradle and the DSL methods

Running the Application

Now, let's run our HelloWorld application. You can either run an Android Virtual Device (which you can set up by using the AVD Manager in Android Studio, as described in the example below) or connect your own Android device through a USB cable.

Setting up an Android device

To run an application from Android Studio on your Android Device, you must enable USB Debugging in the Developer Options in the settings of your device.

Settings > Developer options > USB debugging

If Developer Options is not visible in the settings, navigate to About Phone and tap on the Build Number seven

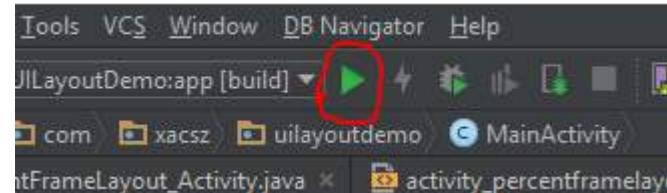
这将使开发者选项出现在你的设置中。

设置>关于手机>版本号

你可能还需要更改build.gradle配置，以便在你的设备支持的版本上构建。

从Android Studio运行

点击Android Studio顶部工具栏的绿色运行按钮。在弹出的窗口中，选择你想运行应用的设备（如有必要，启动一个Android虚拟设备，或参见设置AVD（Android虚拟设备）以了解如何设置），然后点击确定。



在运行Android 4.4 (KitKat) 及可能更高版本的设备上，将显示一个弹出窗口以授权USB 调试。
点击确定以接受。

应用程序现在将安装并运行在您的Android设备或模拟器上。

APK文件位置

当您准备发布应用程序时，您需要配置、构建并测试应用程序的发布版本。

配置任务很简单，涉及基本的代码清理和代码修改任务，有助于优化您的应用程序。构建过程类似于调试构建过程，可以使用JDK和Android SDK工具完成。测试任务作为最终检查，确保您的应用程序在真实环境下按预期运行。准备发布应用程序完成后，您将获得一个签名的APK文件，您可以直接分发给用户，或通过Google Play等应用市场分发。

Android Studio

由于上述示例中使用了Gradle，生成的APK文件位置为：`<您的项目位置>/app/build/outputs/apk/app-debug.apk`

IntelliJ

如果您是IntelliJ用户，在切换到Studio之前直接导入IntelliJ项目，则没有任何变化。输出位置仍然位于：

`out/production/...`

注意：这将在1.0版本左右被弃用

Eclipse

如果你是直接导入Android Eclipse项目，请不要这样做！一旦你的项目中有依赖(jar包或库项目)，这将无法工作，且你的项目不会被正确设置。如果你没有依赖，那么apk文件会在和Eclipse中相同的位置：

`bin/...`

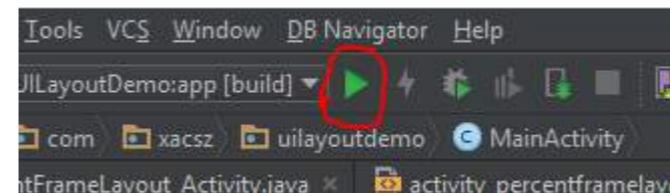
times. This will enable Developer Options to show up in your settings.

Settings > About phone > Build number

You also might need to change build.gradle configuration to build on a version that your device has.

Running from Android Studio

Click the green Run button from the toolbar at the top of Android Studio. In the window that appears, select whichever device you would like to run the app on (start an Android Virtual Device if necessary, or see Setting up an AVD (Android Virtual Device) if you need to set one up) and click OK.



On devices running Android 4.4 (KitKat) and possibly higher, a pop-up will be shown to authorize USB debugging.
Click OK to accept.

The application will now install and run on your Android device or emulator.

APK file location

When you prepare your application for release, you configure, build, and test a release version of your application. The configuration tasks are straightforward, involving basic code cleanup and code modification tasks that help optimize your application. The build process is similar to the debug build process and can be done using JDK and Android SDK tools. The testing tasks serve as a final check, ensuring that your application performs as expected under real-world conditions. When you are finished preparing your application for release you have a signed APK file, which you can distribute directly to users or distribute through an application marketplace such as Google Play.

Android Studio

Since in the above examples Gradle is used, the location of the generated APK file is: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

If you are a user of IntelliJ before switching to Studio, and are importing your IntelliJ project directly, then nothing changed. The location of the output will be the same under:

`out/production/...`

Note: this is will become deprecated sometimes around 1.0

Eclipse

If you are importing Android Eclipse project directly, do not do this! As soon as you have dependencies in your project (jars or Library Projects), this will not work and your project will not be properly setup. If you have no dependencies, then the apk would be under the same location as you'd find it in Eclipse:

`bin/...`

第1.2节：设置Android Studio

Android Studio是Google官方支持和推荐的Android开发集成环境。Android Studio自带了Android SDK管理器，这是一个用于下载开发应用所需的Android SDK组件的工具。

安装Android Studio和Android SDK工具：

1. 下载并安装Android Studio。
2. 通过打开Android Studio，按照
[Android SDK工具更新](#)的说明下载最新的SDK工具和SDK平台工具。你应该安装最新的稳定版本包。

如果你需要处理使用旧版SDK构建的旧项目，可能还需要下载这些旧版本

自Android Studio 2.2起，安装包中捆绑了最新版本的OpenJDK副本，并且这是所有Android Studio项目推荐使用的JDK（Java开发工具包）。这消除了必须安装Oracle JDK包的要求。要使用捆绑的SDK，请按以下步骤操作；

1. 在Android Studio中打开你的项目，并在菜单栏中选择文件>项目结构。
2. 在SDK位置页面的JDK位置下，勾选使用内嵌JDK复选框。
3. 点击确定。

配置Android Studio

Android Studio通过帮助菜单提供访问两个配置文件的入口：

- `studio.vmoptions`：自定义Studio的Java虚拟机(JVM)选项，如堆大小和缓存大小。请注意，在Linux机器上，根据你使用的Android Studio版本，该文件可能被命名为`studio64.vmoptions`。
- `idea.properties`：自定义Android Studio属性，如插件文件夹路径或最大支持的文件大小。

更改/添加主题

你可以根据喜好更改。点击文件->设置->编辑器->颜色&字体->并选择一个主题。你也可以从<http://color-themes.com/>下载新主题。下载.jar或.zip文件后，进入文件->导入设置...并选择下载的文件。

编译应用

在Android Studio中创建一个新项目或打开一个已有项目，然后点击顶部工具栏的绿色播放按钮运行它 。如果按钮是灰色的，你需要等待一会儿，让Android Studio正确索引一些文件，进度可以在底部状态栏看到。

如果你想从命令行创建项目，确保你有一个`local.properties`文件，该文件由Android Studio自动创建。如果需要在没有Android Studio的情况下创建项目，文件中需要有一行以`sdk.dir=`开头，后面跟着你的SDK安装路径。

打开命令行，进入项目目录。输入`./gradlew aR`并按回车。`aR`是`assembleRelease`的快捷方式，它会帮你下载所有依赖并构建应用。最终的APK文件会在`ProjectName/ModuleName/build/outputs/apk`目录下，文件名为`ModuleName-release.apk`。

Section 1.2: Setting up Android Studio

Android Studio is the Android development IDE that is officially supported and recommended by Google. Android Studio comes bundled with the [Android SDK Manager](#), which is a tool to download the Android SDK components required to start developing apps.

Installing Android Studio and Android SDK tools:

1. Download and install [Android Studio](#).
2. Download the latest SDK Tools and SDK Platform-tools by opening the Android Studio, and then following the [Android SDK Tool Updates](#) instructions. You should install the latest available stable packages.

If you need to work on old projects that were built using older SDK versions, you may need to download these versions as well

Since Android Studio 2.2, a copy of the latest OpenJDK comes bundled with the install and is the [recommended JDK](#) (Java Development Kit) for all Android Studio projects. This removes the requirement of having Oracle's JDK package installed. To use the bundled SDK, proceed as follows;

1. Open your project in Android Studio and select **File > Project Structure** in the menu bar.
2. In the **SDK Location** page and under **JDK location**, check the **Use embedded JDK** checkbox.
3. Click **OK**.

Configure Android Studio

Android Studio provides access to two configuration files through the **Help** menu:

- `studio.vmoptions`: Customize options for Studio's Java Virtual Machine (JVM), such as heap size and cache size. Note that on Linux machines this file may be named `studio64.vmoptions`, depending on your version of Android Studio.
- `idea.properties`: Customize Android Studio properties, such as the plugins folder path or maximum supported file size.

Change/add theme

You can change it as your preference. File->Settings->Editor->Colors & Fonts-> and select a theme. Also you can download new themes from <http://color-themes.com/>. Once you have downloaded the .jar .zip file, go to File -> Import Settings... and choose the file downloaded.

Compiling Apps

Create a new project or open an existing project in Android Studio and press the green Play button  on the top toolbar to run it. If it is gray you need to wait a second to allow Android Studio to properly index some files, the progress of which can be seen in the bottom status bar.

If you want to create a project from the shell make sure that you have a `local.properties` file, which is created by Android Studio automatically. If you need to create the project without Android Studio you need a line starting with `sdk.dir=` followed by the path to your SDK installation.

Open a shell and go into the project's directory. Enter `./gradlew aR` and press enter. `aR` is a shortcut for `assembleRelease`, which will download all dependencies for you and build the app. The final APK file will be in `ProjectName/ModuleName/build/outputs/apk` and will be called `ModuleName-release.apk`.

第1.3节：无IDE的Android编程

这是一个极简的Hello World示例，只使用了最基本的Android工具。

需求和假设

- Oracle JDK 1.7或更高版本
- Android SDK工具（仅命令行工具）

此示例假设使用的是Linux。您可能需要根据自己的平台调整语法。

设置 Android SDK

解压SDK发布包后：

1. 使用SDK管理器安装额外的软件包。不要按照指示使用`android update sdk --no-ui`命令捆绑的`Readme.txt`；它会下载大约30 GB的不必要的文件。请改用交互式SDK管理器`android sdk`来获取推荐的最小软件包。
2. 将以下JDK和SDK目录添加到您的执行路径(PATH)中。这是可选的，但以下说明假设您已完成此操作。

- JDK/bin
- SDK/platform-tools
- SDK/tools
- SDK/build-tools/LATEST (如步骤1中安装)

3. 创建一个安卓虚拟设备。使用交互式AVD管理器(`android avd`)。你可能需要调整一下寻求建议；现场指导并不总是有帮助。

(你也可以使用你自己的设备)

4. 运行设备：

```
emulator -avd DEVICE
```

5. 如果设备屏幕显示为锁定状态，请滑动解锁。

在你编写应用程序时保持设备运行。

编写应用程序

0. 切换到一个空的工作目录。

1. 创建源文件：

```
mkdir --parents src/dom/domain  
touch src/dom/domain/SayingHello.java
```

内容：

```
package dom.domain;  
import android.widget.TextView;
```

Section 1.3: Android programming without an IDE

This is a minimalist Hello World example that uses only the most basic Android tools.

Requirements and assumptions

- Oracle JDK 1.7 or later
- Android SDK Tools (just the [command line tools](#))

This example assumes Linux. You may have to adjust the syntax for your own platform.

Setting up the Android SDK

After unpacking the SDK release:

1. Install additional packages using the SDK manager. Don't use `android update sdk --no-ui` as instructed in the bundled `Readme.txt`; it downloads some 30 GB of unnecessary files. Instead use the interactive SDK manager `android sdk` to get the recommended minimum of packages.
2. Append the following JDK and SDK directories to your execution PATH. This is optional, but the instructions below assume it.
 - JDK/bin
 - SDK/platform-tools
 - SDK/tools
 - SDK/build-tools/LATEST (as installed in step 1)
3. Create an Android virtual device. Use the interactive AVD Manager (`android avd`). You might have to fiddle a bit and search for advice; the [on-site instructions](#) aren't always helpful.

(You can also use your own device)

4. Run the device:

```
emulator -avd DEVICE
```

5. If the device screen appears to be locked, then swipe to unlock it.

Leave it running while you code the app.

Coding the app

0. Change to an empty working directory.

1. Make the source file:

```
mkdir --parents src/dom/domain  
touch src/dom/domain/SayingHello.java
```

Content:

```
package dom.domain;  
import android.widget.TextView;
```

```

public final class SayingHello extends android.app.Activity
{
    protected @Override void onCreate( final android.os.Bundle activityState )
    {
        super.onCreate( activityState );
        final TextView textView = new TextView( SayingHello.this );
        textView.setText( "Hello world" );
        setContentView( textView );
    }
}

```

2. 添加一个清单文件：

touch AndroidManifest.xml

内容：

```

<?xml version='1.0'?>
<manifest xmlns:a='http://schemas.android.com/apk/res/android'
package='dom.domain' a:versionCode='0' a:versionName='0'>
    <application a:label='打招呼'>
        <activity a:name='dom.domain.SayingHello'>
            <intent-filter>
                <category a:name='android.intent.category.LAUNCHER' />
                <action a:name='android.intent.action.MAIN' />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

3. 为声明的资源创建一个子目录：

mkdir res

暂时保持为空。

构建代码

0. 生成资源声明的源代码。在此处替换为正确的SDK路径，以及已安装的API版本以进行构建（例如 "android-23"）：

```

aapt package -f \
-I SDK/platforms/android-API/android.jar \
-J src -m \
-M AndroidManifest.xml -S res -v

```

资源声明（下面会进一步说明）实际上是可选的。与此同时，如果 res/ 仍然为空，上述调用不会有任何作用。

1. 将源代码编译为 Java 字节码（java → .class）：

```

javac \
-bootclasspath SDK/platforms/android-API/android.jar \
-classpath src -source 1.7 -target 1.7 \
src/dom/domain/*.java

```

```

public final class SayingHello extends android.app.Activity
{
    protected @Override void onCreate( final android.os.Bundle activityState )
    {
        super.onCreate( activityState );
        final TextView textView = new TextView( SayingHello.this );
        textView.setText( "Hello world" );
        setContentView( textView );
    }
}

```

2. Add a manifest:

touch AndroidManifest.xml

Content:

```

<?xml version='1.0'?>
<manifest xmlns:a='http://schemas.android.com/apk/res/android'
package='dom.domain' a:versionCode='0' a:versionName='0'>
    <application a:label='Saying hello'>
        <activity a:name='dom.domain.SayingHello'>
            <intent-filter>
                <category a:name='android.intent.category.LAUNCHER' />
                <action a:name='android.intent.action.MAIN' />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

3. Make a sub-directory for the declared resources:

mkdir res

Leave it empty for now.

Building the code

0. Generate the source for the resource declarations. Substitute here the correct path to your **SDK**, and the installed **API** to build against (e.g. "android-23"):

```

aapt package -f \
-I SDK/platforms/android-API/android.jar \
-J src -m \
-M AndroidManifest.xml -S res -v

```

Resource declarations (described further below) are actually optional. Meantime the above call does nothing if res/ is still empty.

1. Compile the source code to Java bytecode (java → .class):

```

javac \
-bootclasspath SDK/platforms/android-API/android.jar \
-classpath src -source 1.7 -target 1.7 \
src/dom/domain/*.java

```

2. 将字节码从 Java 转换为 Android 格式 (.class → .dex) :

首先使用 Jill (.class → .jayce) :

```
java -jar SDK/build-tools/LATEST/jill.jar \
--output classes.jayce src
```

然后使用 Jack (.jayce → .dex) :

```
java -jar SDK/build-tools/LATEST/jack.jar \
--import classes.jayce --output-dex .
```

Android 字节码曾被称为“Dalvik 可执行代码”，简称“dex”。

如果愿意，你可以用一次调用 Jack 来替代第11步和第12步；它可以直接从 Java 源代码编译 (.java → .dex)。但使用 javac 编译有其优势。它是一个更为知名、文档更完善且应用更广泛的工具。

3. 打包资源文件，包括清单文件：

```
aapt package -f \
-F app.apkPart \
-I SDK/platforms/android-API/android.jar \
-M AndroidManifest.xml -S res -v
```

这会生成一个部分 APK 文件 (Android 应用包)。

4. 使用 ApkBuilder 工具制作完整的 APK：

```
java -classpath SDK/tools/lib/sdklib.jar \
com.android.sdklib.build.ApkBuilderMain \
app.apkUnalign \
-d -f classes.dex -v -z app.apkPart
```

它会警告，“此工具已弃用。更多信息请参见 --help。”如果 --help 命令失败，`ArrayIndexOutOfBoundsException`，改为不传递任何参数：

```
java -classpath SDK/tools/lib/sdklib.jar \
com.android.sdklib.build.ApkBuilderMain
```

它解释了命令行工具 (ApkBuilderMain) 已被弃用，建议直接调用Java API (ApkBuilder)。
(如果你知道如何从命令行操作，请更新此示例。)

5. 优化APK的数据对齐（推荐做法）：

```
zipalign -f -v 4 app.apkUnalign app.apk
```

安装和运行

0. 将应用安装到Android设备：

2. Translate the bytecode from Java to Android (.class → .dex):

First using Jill (.class → .jayce):

```
java -jar SDK/build-tools/LATEST/jill.jar \
--output classes.jayce src
```

Then Jack (.jayce → .dex):

```
java -jar SDK/build-tools/LATEST/jack.jar \
--import classes.jayce --output-dex .
```

Android bytecode used to be called "Dalvik executable code", and so "dex".

You could replace steps 11 and 12 with a single call to Jack if you like; it can compile directly from Java source (.java → .dex). But there are advantages to compiling with javac. It's a better known, better documented and more widely applicable tool.

3. Package up the resource files, including the manifest:

```
aapt package -f \
-F app.apkPart \
-I SDK/platforms/android-API/android.jar \
-M AndroidManifest.xml -S res -v
```

That results in a partial APK file (Android application package).

4. Make the full APK using the ApkBuilder tool:

```
java -classpath SDK/tools/lib/sdklib.jar \
com.android.sdklib.build.ApkBuilderMain \
app.apkUnalign \
-d -f classes.dex -v -z app.apkPart
```

It warns, "THIS TOOL IS DEPRECATED. See --help for more information." If --help fails with an `ArrayIndexOutOfBoundsException`, then instead pass no arguments:

```
java -classpath SDK/tools/lib/sdklib.jar \
com.android.sdklib.build.ApkBuilderMain
```

It explains that the CLI (ApkBuilderMain) is deprecated in favour of directly calling the Java API (ApkBuilder).
(If you know how to do that from the command line, please update this example.)

5. Optimize the data alignment of the APK ([recommended practice](#)):

```
zipalign -f -v 4 app.apkUnalign app.apk
```

Installing and running

0. Install the app to the Android device:

```
adb install -r app.apk
```

1. 启动应用：

```
adb shell am start -n dom.domain/.SayHello
```

它应该运行并说你好。

就是这样。使用基本的Android工具说“你好”就是这么简单。

声明资源

本节是可选的。资源声明对于一个简单的“你好，世界”应用程序来说不是必需的。如果你的应用程序也需要，那么你可以通过省略第10步，并从第13步中移除对res/目录的引用，来简化构建过程。

否则，下面是一个简短的示例，说明如何声明资源以及如何引用它。

0. 添加资源文件：

```
mkdir res/values  
touch res/values/values.xml
```

内容：

```
<?xml version='1.0'?>  
<resources>  
    <string name='appLabel'>Saying hello</string>  
</resources>
```

1. 从 XML 清单中引用资源。这是一种声明式的引用方式：

```
<!-- <application a:label='Saying hello'> -->  
    <application a:label='@string/appLabel'>
```

2. 从Java源代码中引用相同的资源。这是一个强制引用：

```
// v.setText( "Hello world" );  
v.setText( "这个应用叫做 "  
    +getResources().getString( R.string.appLabel ) );
```

3. 通过重建、重新安装并重新运行应用（步骤10-17）来测试上述修改。

它应该会重新启动并显示，“这个应用叫做 Saying hello”。

卸载应用

```
adb uninstall dom.domain
```

另请参见

- [原始问题](#) - 促使此示例的原始问题
- [工作示例](#) - 使用上述命令的可运行构建脚本

```
adb install -r app.apk
```

1. Start the app:

```
adb shell am start -n dom.domain/.SayHello
```

It should run and say hello.

That's all. That's what it takes to say hello using the basic Android tools.

Declaring a resource

This section is optional. Resource declarations aren't required for a simple "hello world" app. If they aren't required for your app either, then you could streamline the build somewhat by omitting step 10, and removing the reference to the res/ directory from step 13.

Otherwise, here's a brief example of how to declare a resource, and how to reference it.

0. Add a resource file:

```
mkdir res/values  
touch res/values/values.xml
```

Content:

```
<?xml version='1.0'?>  
<resources>  
    <string name='appLabel'>Saying hello</string>  
</resources>
```

1. Reference the resource from the XML manifest. This is a declarative style of reference:

```
<!-- <application a:label='Saying hello'> -->  
    <application a:label='@string/appLabel'>
```

2. Reference the same resource from the Java source. This is an imperative reference:

```
// v.setText( "Hello world" );  
v.setText( "This app is called "  
    + getResources().getString( R.string.appLabel ) );
```

3. Test the above modifications by rebuilding, reinstalling and re-running the app (steps 10-17).

It should restart and say, "This app is called Saying hello".

Uninstalling the app

```
adb uninstall dom.domain
```

See also

- [original question](#) - The original question that prompted this example
- [working example](#) - A working build script that uses the above commands

第1.4节：应用基础

Android应用程序是用Java编写的。Android SDK工具将代码、数据和资源文件编译成APK (Android包)。通常，一个APK文件包含应用的所有内容。

每个应用运行在其自己的虚拟机 (VM) 上，因此应用可以与其他应用隔离运行。Android系统遵循最小权限原则。每个应用仅能访问其完成工作所需的组件，且不多于此。然而，应用之间可以通过共享Linux用户ID等方式共享数据，或者应用可以请求权限访问设备数据，如SD卡、联系人等。

应用组件

应用组件是Android应用的构建模块。每个组件在Android应用中扮演特定角色，具有独特的用途和生命周期（组件创建和销毁的流程和时间）。以下是四种应用组件：

1. 活动 (Activities)：活动代表带有用户界面 (UI) 的单个屏幕。一个Android应用可能有多个活动 (例如，一个电子邮件应用可能有一个活动用于列出所有邮件，另一个用于显示每封邮件的内容，另一个用于撰写新邮件)。应用中的所有活动协同工作，以创建用户体验 (UX)。
2. 服务 (Services)：服务在后台运行，以执行长时间运行的操作或为其他组件执行工作远程进程。服务不提供任何用户界面，仅在后台运行，依赖用户输入。(例如，一个服务可以在用户使用其他应用时在后台播放音乐，或者它可以从互联网下载数据而不会阻塞用户与安卓设备的交互。)
3. 内容提供者：内容提供者管理共享的应用数据。有四种方式可以在一个应用程序：它可以写入文件并存储在文件系统中，插入或更新到 SQLite 数据库，发布到网络，或保存到应用程序可以访问的任何其他持久存储位置。通过内容提供者，其他应用程序可以查询甚至修改数据。(例如，Android 系统提供了一个内容提供者，管理用户的联系信息，以便任何具有权限的应用程序都可以查询联系人。内容提供者也可以用于保存应用程序私有的数据，以提高数据完整性。)
4. 广播接收器：广播接收器响应系统范围内的广播通知 (例如广播通知 (例如屏幕已关闭、电池电量低等) 或来自应用程序 (例如通知其他应用某些数据已下载到设备并可供使用))。广播接收器没有用户界面，但可以在状态栏显示通知以提醒用户。通常，广播接收器用作应用程序其他组件的入口，主要包括活动和服务。

Android系统的一个独特之处在于，任何应用都可以启动另一个应用的组件 (例如，如果你想拨打电话、发送短信、打开网页或查看照片，已经有应用可以完成这些操作，你的应用可以利用它，而不必为同一任务开发新的活动)。

当系统启动一个组件时，它会启动该应用的进程 (如果该进程尚未运行，即在Android系统上每个应用在任何给定时间只能运行一个前台进程)，并实例化该组件所需的类。因此，该组件运行在其所属应用的进程中。因此，与其他系统上的应用不同，Android应用没有单一的入口点 (没有main()方法)。

由于系统在单独的进程中运行每个应用程序，一个应用程序无法直接激活另一个应用程序的组件，但Android系统可以。因此，要启动另一个应用程序的组件，一个应用程序必须向系统发送一条消息，指定启动该组件的意图，然后系统将启动该组件。

背景

类`android.content.Context`的实例提供了与执行应用程序的Android系统的连接。需要`Context`的实例来访问项目的资源和全局信息

Section 1.4: Application Fundamentals

Android Apps are written in Java. The Android SDK tools compile the code, data and resource files into an APK (Android package). Generally, one APK file contains all the content of the app.

Each app runs on its own virtual machine(VM) so that app can run isolated from other apps. Android system works with the principle of least privilege. Each app only has access to the components which it requires to do its work, and no more. However, there are ways for an app to share data with other apps, such as by sharing Linux user id between app, or apps can request permission to access device data like SD card, contacts etc.

App Components

App components are the building blocks of an Android app. Each components plays a specific role in an Android app which serves a distinct purpose and has distinct life-cycles(the flow of how and when the component is created and destroyed). Here are the four types of app components:

1. **Activities:** An activity represents a single screen with a User Interface(UI). An Android app may have more than one activity. (e.g. an email app might have one activity to list all the emails, another to show the contents of each email, and another to compose new email.) All the activities in an App work together to create a User eXperience (UX).
2. **Services:** A service runs in the background to perform long-running operations or to perform work for a remote processes. A service does not provide any UI, it runs only in the background with the User's input. (e.g. a service can play music in the background while the user is in a different App, or it might download data from the internet without blocking user's interaction with the Android device.)
3. **Content Providers:** A content provider manages shared app data. There are four ways to store data in an app: it can be written to a file and stored in the file system, inserted or updated to a SQLite database, posted to the web, or saved in any other persistent storage location the App can access. Through content providers, other Apps can query or even modify the data. (e.g. Android system provides a content provider that manages the user's contact information so that any app which has permission can query the contacts.) Content providers can also be used to save the data which is private to the app for better data integrity.
4. **Broadcast receivers:** A broadcast receiver responds to the system-wide broadcasts of announcements (e.g. a broadcast announcing that the screen has turned off, the battery is low, etc.) or from Apps (e.g. to let other apps know that some data has been downloaded to the device and is available for them to use). Broadcast receivers don't have UIs but they can show notification in the status bar to alert the user. Usually broadcast receivers are used as a gateway to other components of the app, consisting mostly of activities and services.

One unique aspect of the Android system is that any app can start another app's component (e.g. if you want to make call, send SMS, open a web page, or view a photo, there is an app which already does that and your app can make use of it, instead of developing a new activity for the same task).

When the system starts a component, it starts the process for that app (if it isn't already running, i.e. only one foreground process per app can run at any given time on an Android system) and instantiates the classes needed for that component. Thus the component runs on the process of that App that it belongs to. Therefore, unlike apps on other systems, Android apps don't have a single entry point (there is no `main()` method).

Because the system runs each app in a separate process, one app cannot directly activate another app's components, however the Android system can. Thus to start another app's component, one app must send a message to the system that specifies an intent to start that component, then the system will start that component.

Context

Instances of the class `android.content.Context` provide the connection to the Android system which executes the application. Instance of Context is required to get access to the resources of the project and the global information

关于应用程序环境的信息。

让我们来看一个易于理解的例子：假设你在一家酒店，想吃点东西。你打电话给客房服务，叫他们给你送东西或帮你打扫。现在把这家酒店看作一个Android应用程序，你自己是一个活动（Activity），而客房服务人员就是你的上下文（Context），它为你提供访问酒店资源（如客房服务、食物等）的途径。

再举一个例子，你在餐厅坐在桌子旁，每张桌子都有一名服务员，每当你想点餐时，你会请服务员帮你点。服务员会帮你下单，食物随后送到你的桌上。在这个例子中，餐厅是一个Android应用程序，桌子或顾客是应用组件，食物是你的应用资源，而服务员就是你的上下文，提供了访问食物等资源的方式。

激活上述任何组件都需要上下文的实例。不仅如此，几乎所有系统资源：使用视图创建用户界面（稍后讨论）、创建系统服务实例、启动新的活动或服务——都需要上下文。

更详细的描述写在这里。

第1.5节：设置AVD（Android虚拟设备）

简而言之 它基本上允许我们模拟真实设备并在没有真实设备的情况下测试应用程序。

根据Android开发者文档，

Android虚拟设备（AVD）定义让你定义想要在Android模拟器中模拟的Android手机、平板、Android Wear或Android TV设备的特性。AVD管理器帮助你轻松创建和管理AVD。

要设置一个AVD，请按照以下步骤操作：

1.点击此按钮以打开 AVD 管理器：



2.你应该会看到如下对话框：

about the app's environment.

Let's have an easy to digest example: Consider you are in a hotel, and you want to eat something. You call room-service and ask them to bring you things or clean up things for you. Now think of this hotel as an Android app, yourself as an activity, and the room-service person is then your context, which provides you access to the hotel resources like room-service, food items etc.

Yet another example, You are in a restaurant sitting on a table, each table has an attendant, whenever you want to order food items you ask the attendant to do so. The attendant then places your order and your food items get served on your table. Again in this example, the restaurant is an Android App, the tables or the customers are App components, the food items are your App resources and the attendant is your context thus giving you a way to access the resources like food items.

Activating any of the above components requires the context's instance. Not just only the above, but almost every system resource: creation of the UI using views(discussed later), creating instance of system services, starting new activities or services -- all require context.

More detailed description is written [here](#).

Section 1.5: Setting up an AVD (Android Virtual Device)

TL;DR It basically allows us to simulate real devices and test our apps without a real device.

According to [Android Developer Documentation](#),

an **Android Virtual Device (AVD) definition** lets you define the characteristics of an Android Phone, Tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator. The AVD Manager helps you easily create and manage AVDs.

To set up an AVD, follow these steps:

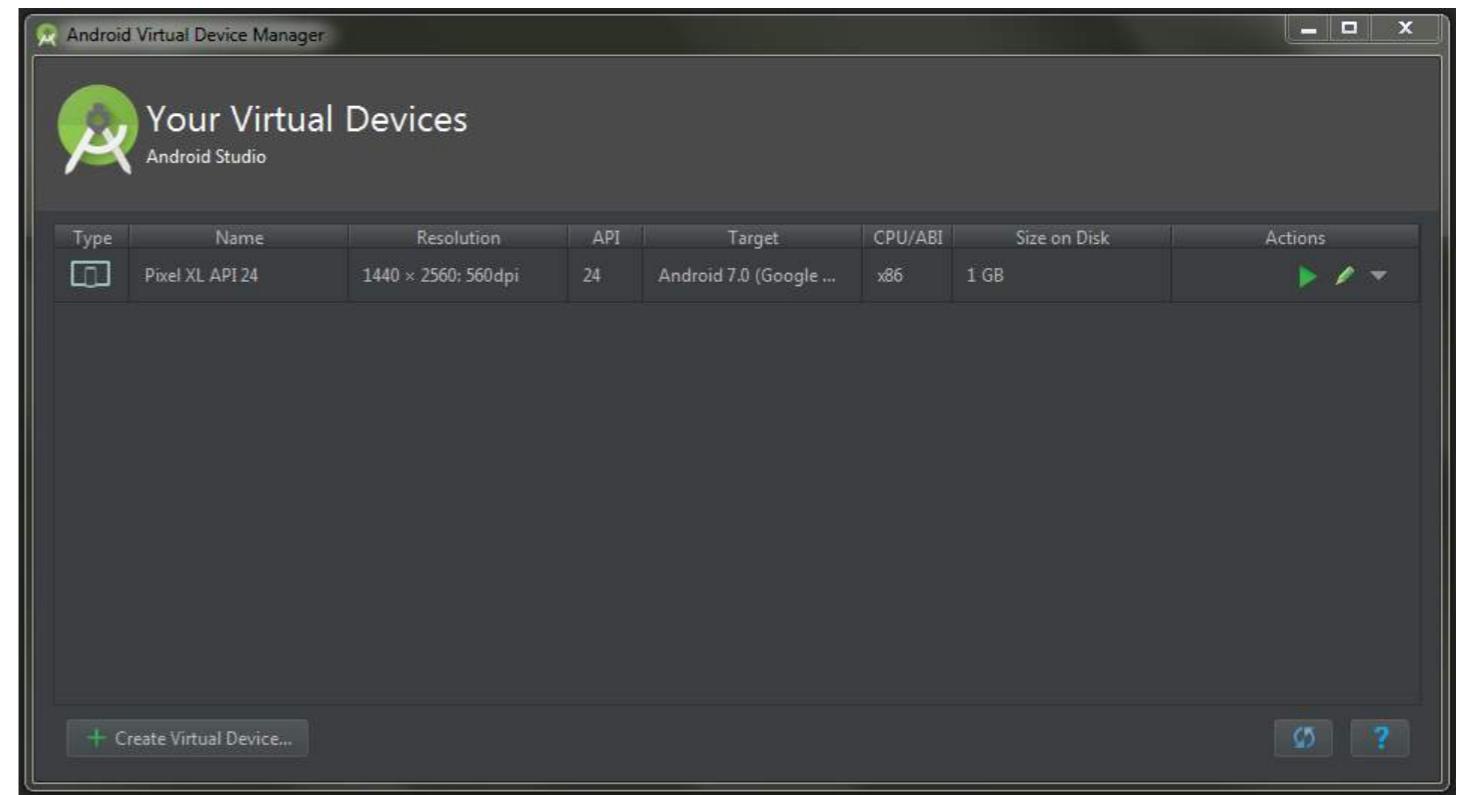
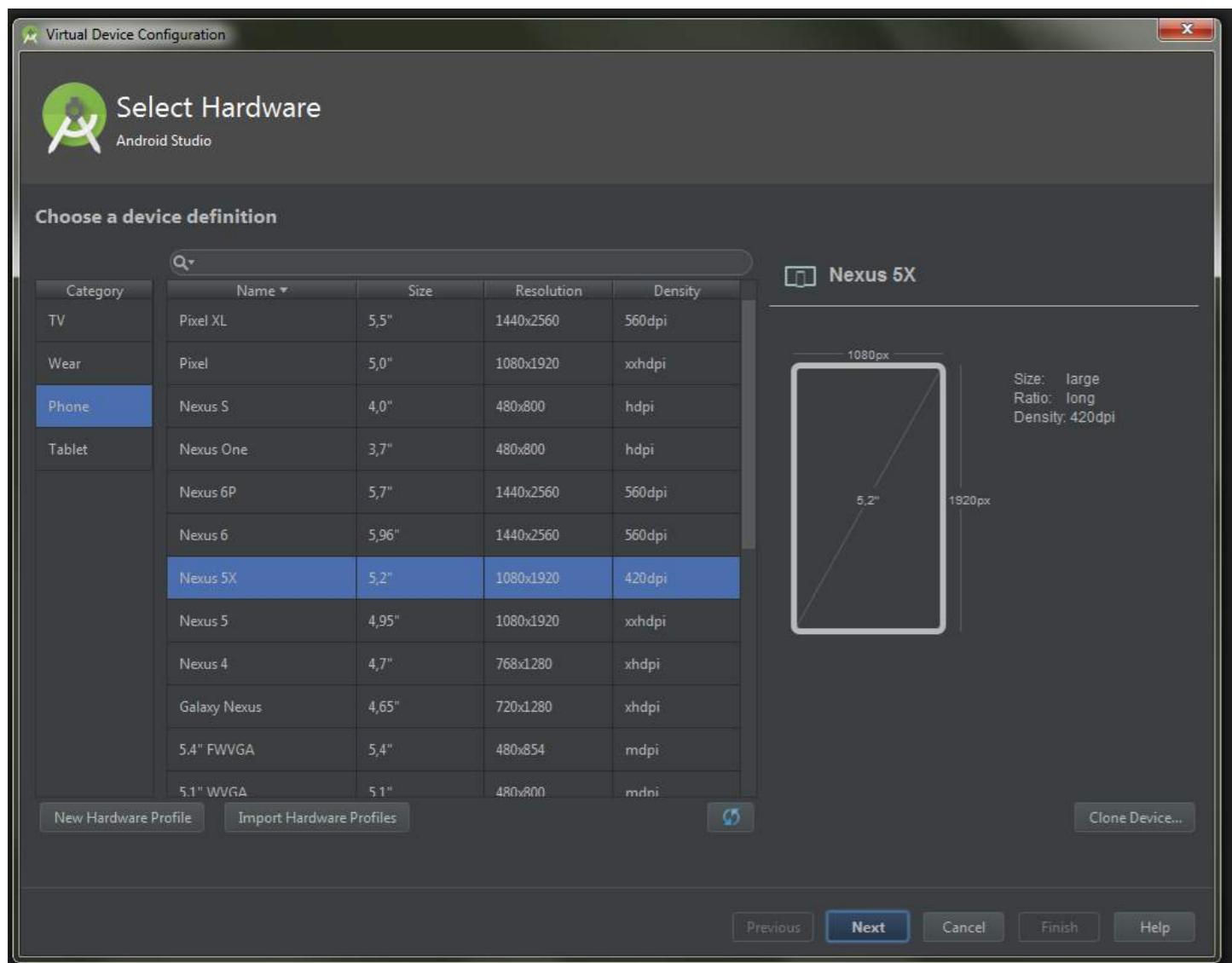
1. Click this button to bring up the AVD Manager:



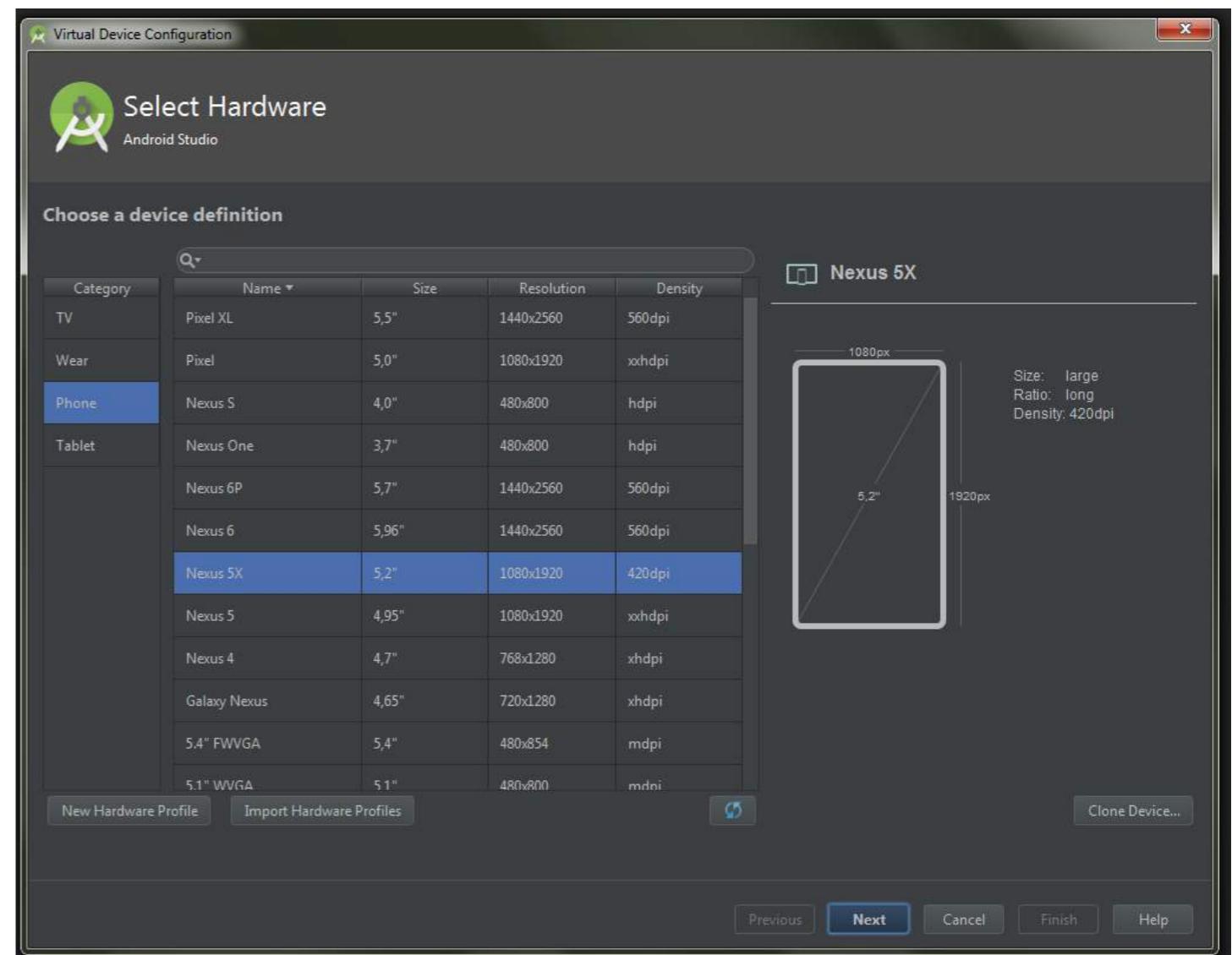
2. You should see a dialog like this:



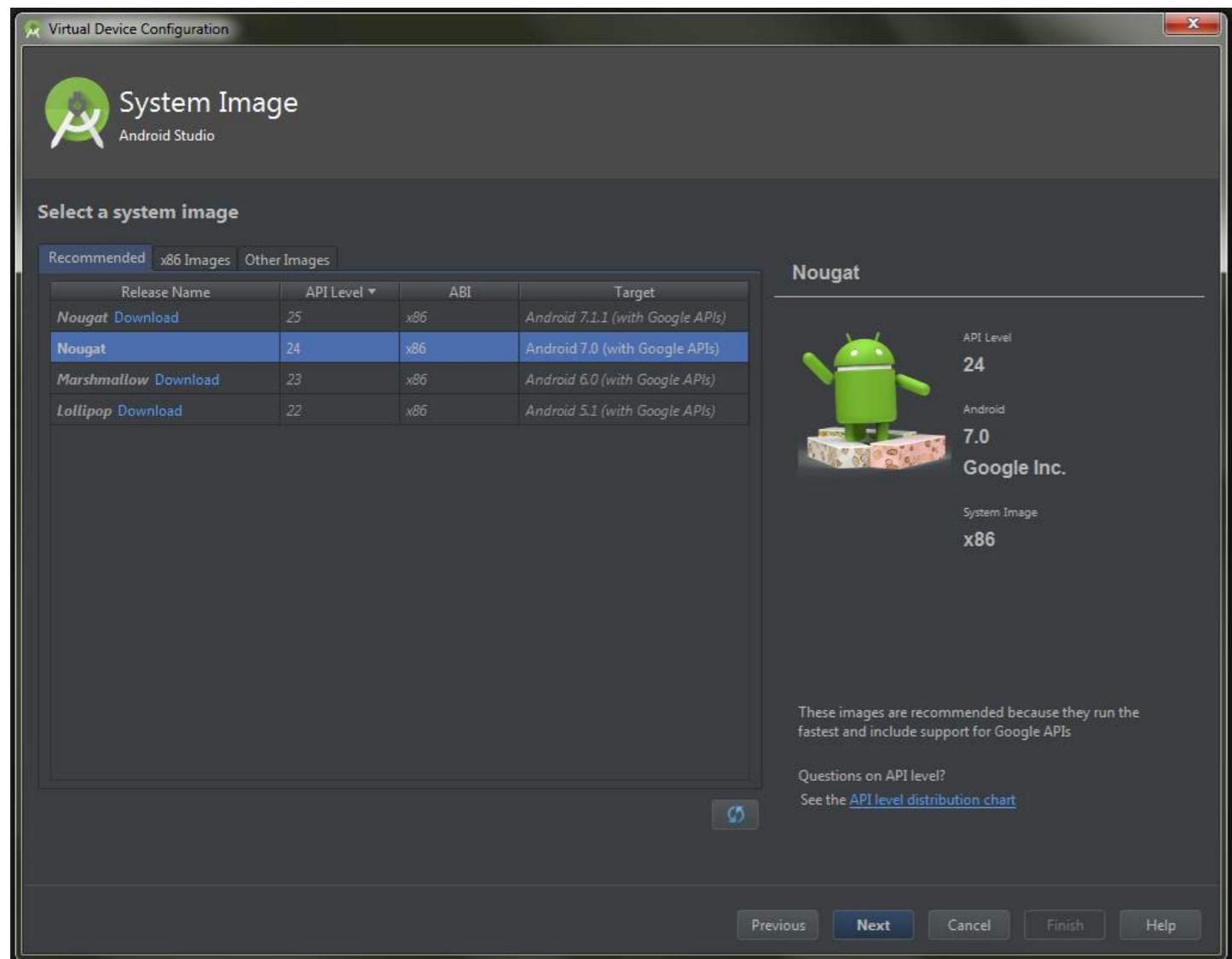
3.现在点击+创建虚拟设备...按钮。这将打开虚拟设备配置对话框：



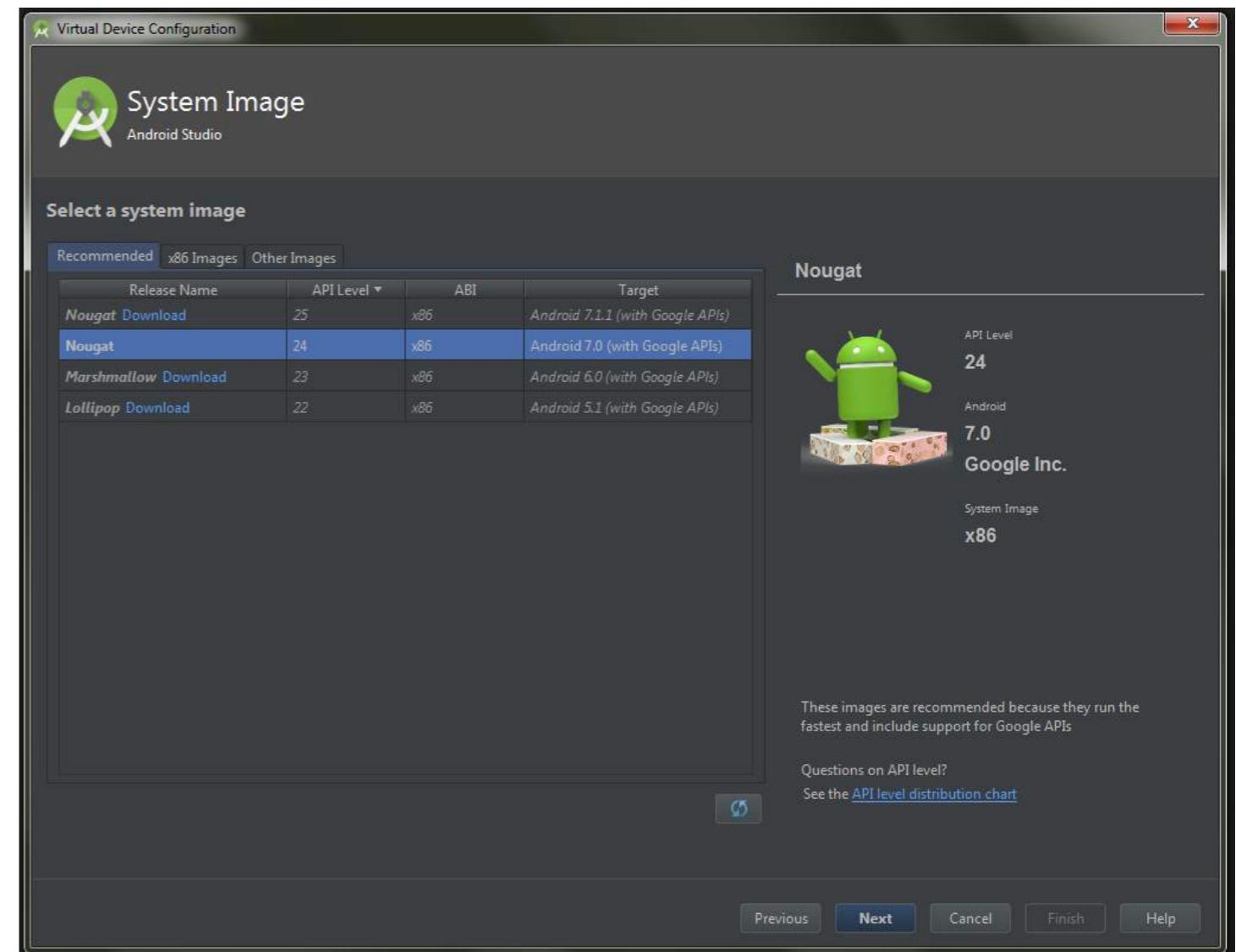
3. Now click the + Create Virtual Device... button. This will bring up Virtual Device Configuration Dialog:



4. 选择你想要的任何设备，然后点击下一步：

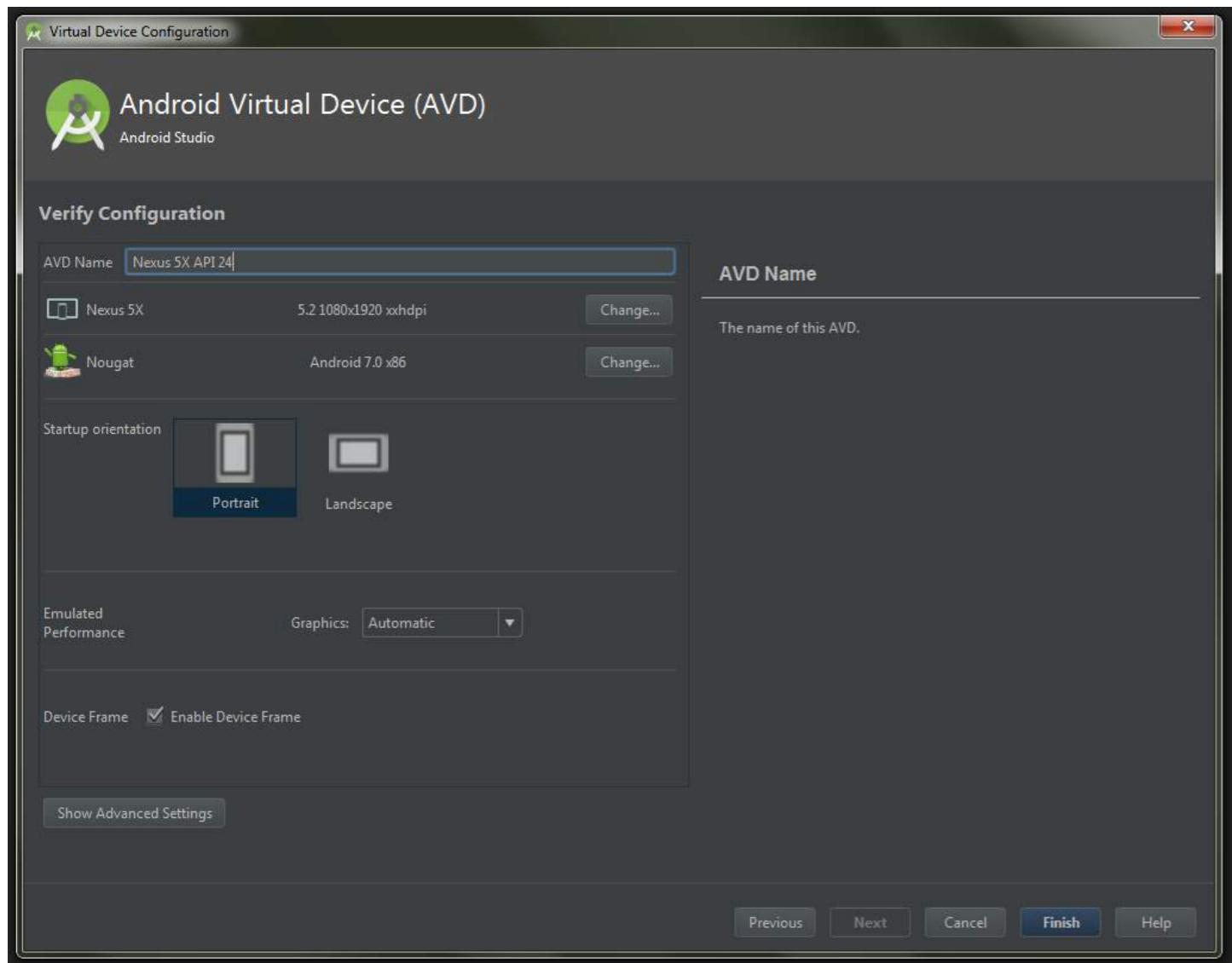


4. Select any device you want, then click Next:



5. 这里你需要为模拟器选择一个 Android 版本。你可能还需要先通过点击下载来下载它。选择版本后，点击下一步。

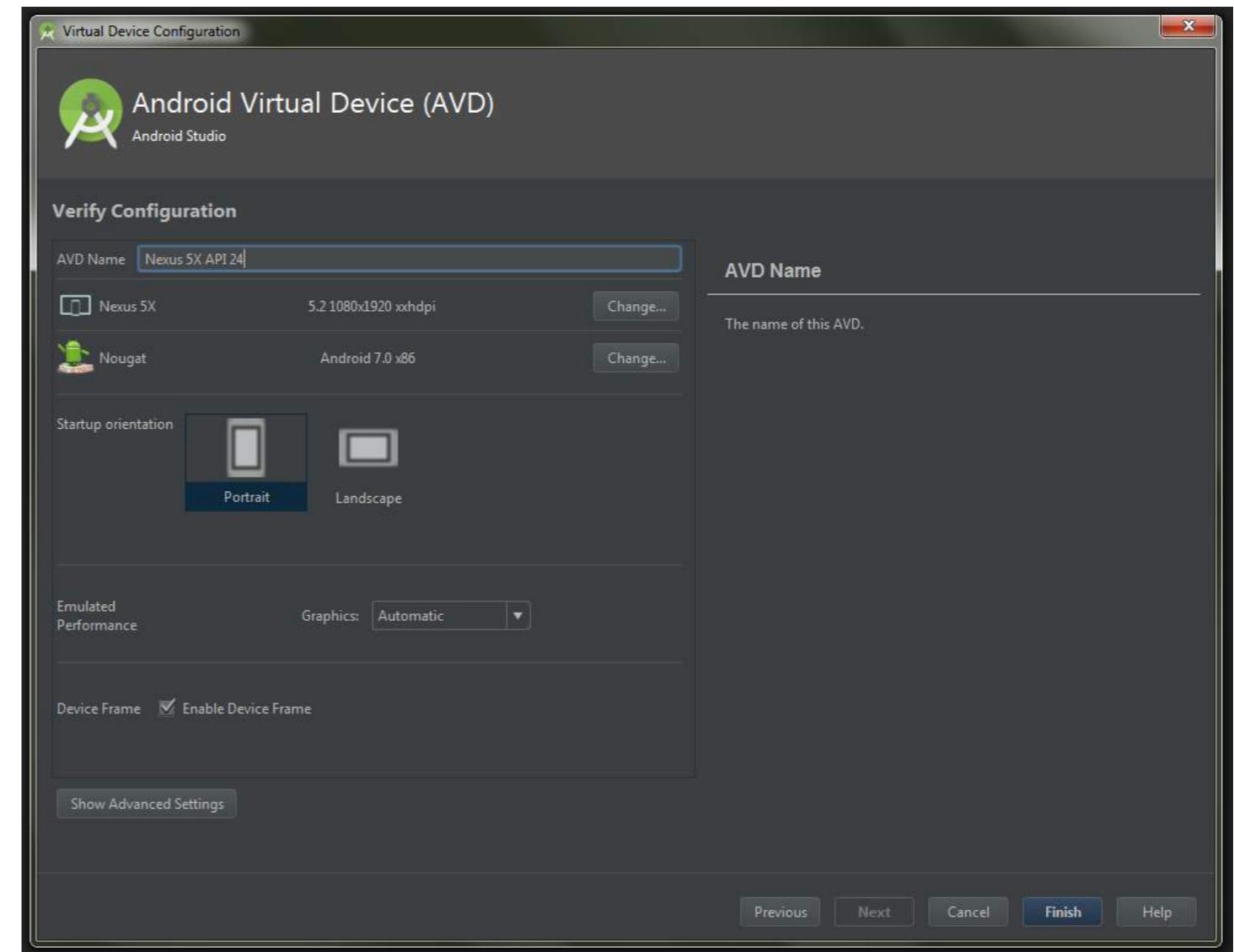
5. Here you need to choose an Android version for your emulator. You might also need to download it first by clicking Download. After you've chosen a version, click Next.



6. 在这里，输入你的模拟器名称、初始方向，以及是否显示边框。
选择完这些后，点击完成。

7. 现在你已经有了一个新的AVD，可以在上面启动你的应用程序。

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
□	Nexus 5X API 24	1080 × 1920: 420dpi	24	Android 7.0 (Google ...)	x86	650 MB	▶️ 🖊️ ⚙️



6. Here, enter a name for your emulator, initial orientation, and whether you want to display a frame around it.
After you chosen all these, click Finish.

7. Now you got a new AVD ready for launching your apps on it.

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
□	Nexus 5X API 24	1080 × 1920: 420dpi	24	Android 7.0 (Google ...)	x86	650 MB	▶️ 🖊️ ⚙️

第2章：Android Studio

第2.1节：设置Android Studio

系统要求

- Microsoft® Windows® 8/7/Vista/2003 (32位或64位)。
- Mac® OS X® 10.8.5或更高版本，最高至10.9 (Mavericks)
- GNOME或KDE桌面环境

安装

Windows

1. 下载并安装JDK (Java开发工具包) 版本8
2. [下载Android Studio](#)
3. 启动Android Studio.exe然后指定JDK路径并下载最新的SDK

Linux

1. 下载并安装JDK (Java开发工具包) 版本8
2. [下载Android Studio](#)
3. 解压zip文件
4. 打开终端，切换到解压后的文件夹，进入bin目录（例如cd android-studio/bin）
5. 运行 ./studio.sh

第2.2节：在Android Studio中查看和添加快捷键

通过进入设置 >> 键位映射，会弹出一个窗口显示所有编辑器操作及其名称和快捷键。有些编辑器操作没有快捷键。对此右键点击并添加新的快捷键。

查看下图

Chapter 2: Android Studio

Section 2.1: Setup Android Studio

System Requirements

- Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit).
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
- GNOME or KDE desktop

Installation

Window

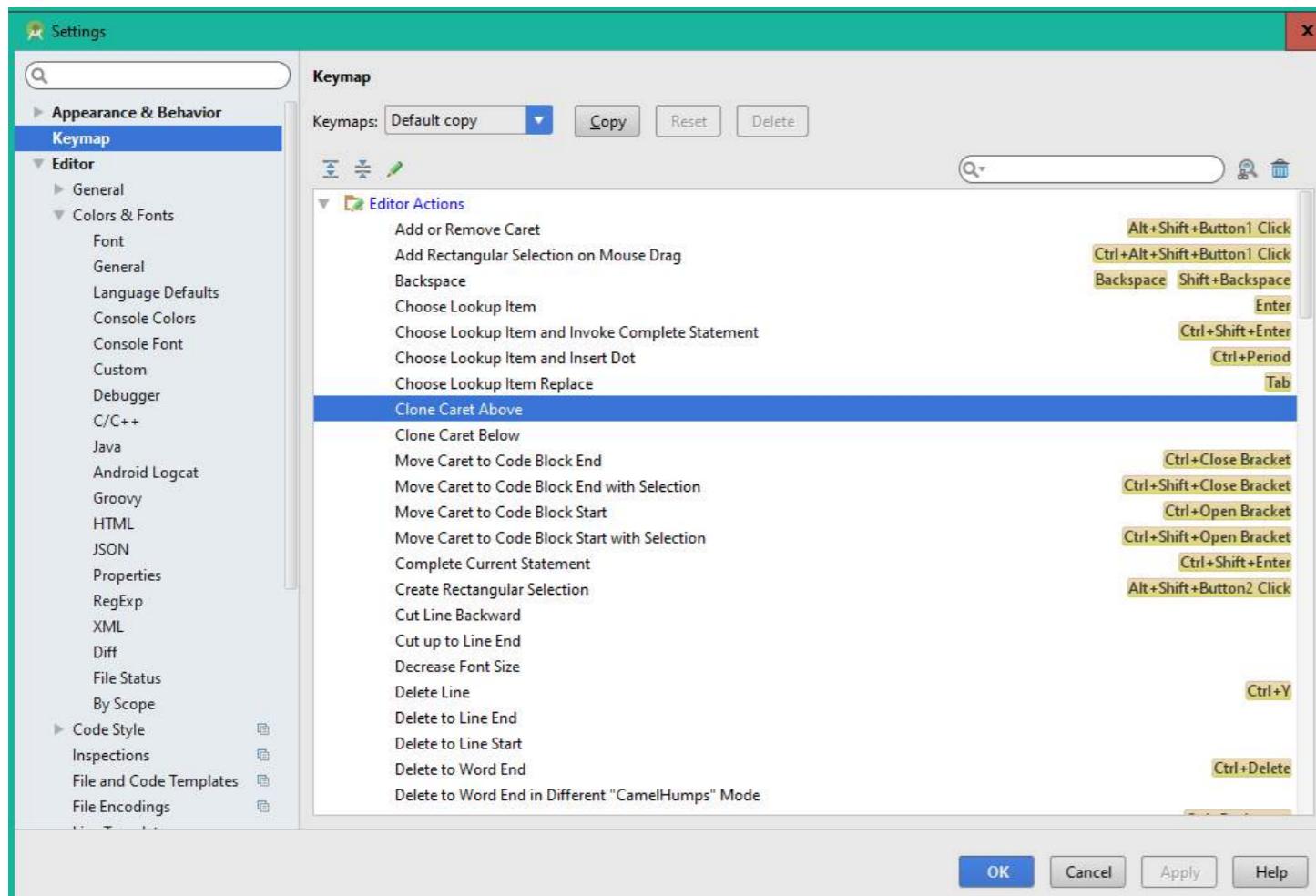
1. Download and install [JDK \(Java Development Kit\)](#) version 8
2. Download [Android Studio](#)
3. Launch `Android Studio.exe` then mention JDK path and download the latest SDK

Linux

1. Download and install [JDK \(Java Development Kit\)](#) version 8
2. Download [Android Studio](#)
3. Extract the zip file
4. Open terminal, cd to the extracted folder, cd to bin (example `cd android-studio/bin`)
5. Run `./studio.sh`

Section 2.2: View And Add Shortcuts in Android Studio

By going to Settings >> Keymap A window will popup showing All the Editor Actions with their name and shortcuts. Some of the Editor Actions do not have shortcuts. So right click on that and add a new shortcut to that. Check the image below



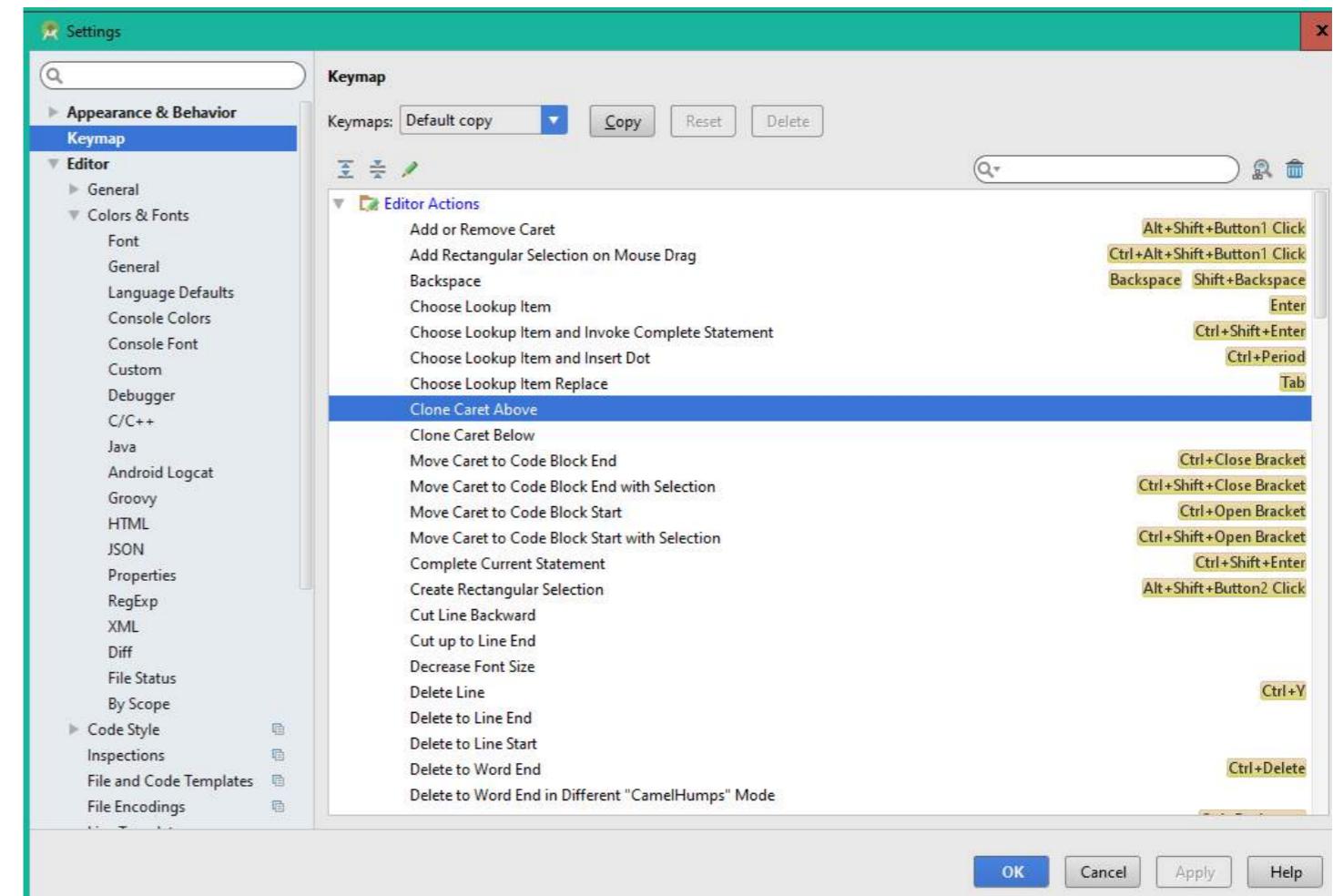
第2.3节：Android Studio常用快捷键

下面是一些更常用/实用的快捷键。

这些基于默认的IntelliJ快捷键映射。你可以通过文件切换到其他常用IDE快捷键映射

-> 设置 -> 键位映射 -> <从键位映射下拉菜单中选择 Eclipse/Visual Studio/等>

操作	快捷键
格式化代码	CTRL + ALT + L
添加未实现的方法	CTRL + I
显示日志	ALT + 6
构建	CTRL + F9
构建并运行	CTRL + F10
查找	CTRL + F
在项目中查找	CTRL + SHIFT + F
查找和替换	CTRL + R
在项目中查找和替换	CTRL + SHIFT + R
重写方法	CTRL + O
显示项目	ALT + 1
隐藏项目 - logcat	SHIFT + ESC
全部折叠	CTRL + SHIFT + 数字键盘 +
查看调试点	CTRL + SHIFT + F8
展开全部	CTRL + SHIFT + 数字键盘 -
打开设置	ALT + S



Section 2.3: Android Studio useful shortcuts

The following are some of the more common/useful shortcuts.

These are based on the default IntelliJ shortcut map. You can switch to other common IDE shortcut maps via [File](#)

-> [Settings](#) -> [Keymap](#) -> <Choose Eclipse/Visual Studio/etc from Keystrokes dropdown>

Action	Shortcut
Format code	CTRL + ALT + L
Add unimplemented methods	CTRL + I
Show logcat	ALT + 6
Build	CTRL + F9
Build and Run	CTRL + F10
Find	CTRL + F
Find in project	CTRL + SHIFT + F
Find and replace	CTRL + R
Find and replace in project	CTRL + SHIFT + R
Override methods	CTRL + O
Show project	ALT + 1
Hide project - logcat	SHIFT + ESC
Collapse all	CTRL + SHIFT + NumPad +
View Debug Points	CTRL + SHIFT + F8
Expand all	CTRL + SHIFT + NumPad -
Open Settings	ALT + S

选择目标（在项目视图中打开当前文件）

操作 **快捷键**

ALT + F1 → 回车
SHIFT → SHIFT (双击 Shift)
CTRL → ALT + T
ALT + CTRL

全局搜索

代码 | 包裹

从选中代码创建方法

重构：

操作
重构此项（当前元素所有适用重构操作的菜单/选择器）

快捷键
Mac **CTRL** + **T** - Win/Linux **CTRL** + **ALT** + **T**

重命名

SHIFT + **F6**

提取方法

Mac **命令** + **ALT** + **M** - Win/Linux **CTRL** + **ALT** + **M**

提取参数

Mac **命令** + **ALT** + **P** - Win/Linux **CTRL** + **ALT** + **P**

提取变量

Mac **命令** + **ALT** + **V** - Win/Linux **CTRL** + **ALT** + **V**

Select Target (open current file in Project view) **ALT** + **F1** → **ENTER**

Search Everywhere

Code | Surround With

Create method from selected code

操作
SHIFT → **SHIFT** (Double shift)
CTRL → **ALT** + **T**
ALT + **CTRL**

Refactor:

Action
Refactor This (menu/picker for all applicable refactor actions of the current element)

Rename

Extract Method

Extract Parameter

Extract Variable

Shortcut
Mac **CTRL** + **T** - Win/Linux **CTRL** + **ALT** + **T**

SHIFT + **F6**

Mac **CMD** + **ALT** + **M** - Win/Linux **CTRL** + **ALT** + **M**

Mac **CMD** + **ALT** + **P** - Win/Linux **CTRL** + **ALT** + **P**

Mac **CMD** + **ALT** + **V** - Win/Linux **CTRL** + **ALT** + **V**

第2.4节：Android Studio 性能提升技巧

启用离线工作：

1. 点击 文件 -> 设置。搜索“gradle”，并勾选离线工作框。
2. 转到编译器（在同一设置对话框中，位于Gradle下方），并在命令行选项文本框中添加--offline 框。

提升Gradle性能

在你的gradle.properties文件中添加以下两行代码。

```
org.gradle.daemon=true  
org.gradle.parallel=true
```

增加studio.vmoptions文件中-Xmx和-Xms的值

```
-Xms1024m  
-Xmx4096m  
-XX:MaxPermSize=1024m  
-XX:ReservedCodeCacheSize=256m  
-XX:+UseCompressedOops
```

Windows

```
%USERPROFILE%.{FOLDER_NAME}\studio.exe.vmoptions 和或  
%USERPROFILE%.{FOLDER_NAME}\studio64.exe.vmoptions
```

Mac

```
~/Library/Preferences/{FOLDER_NAME}/studio.vmoptions
```

Linux

```
~/.{FOLDER_NAME}/studio.vmoptions 和或 ~/.{FOLDER_NAME}/studio64.vmoptions
```

Section 2.4: Android Studio Improve performance tip

Enable Offline Work:

1. Click File -> Settings. Search for "gradle" and click in Offline work box.
2. Go to Compiler (in same settings dialog just below Gradle) and add --offline to Command-line Options text box.

Improve Gradle Performance

Add following two line of code in your gradle.properties file.

```
org.gradle.daemon=true  
org.gradle.parallel=true
```

Increasing the value of -Xmx and -Xms in studio.vmoptions file

```
-Xms1024m  
-Xmx4096m  
-XX:MaxPermSize=1024m  
-XX:ReservedCodeCacheSize=256m  
-XX:+UseCompressedOops
```

Window

```
%USERPROFILE%.{FOLDER_NAME}\studio.exe.vmoptions and/or  
%USERPROFILE%.{FOLDER_NAME}\studio64.exe.vmoptions
```

Mac

```
~/Library/Preferences/{FOLDER_NAME}/studio.vmoptions
```

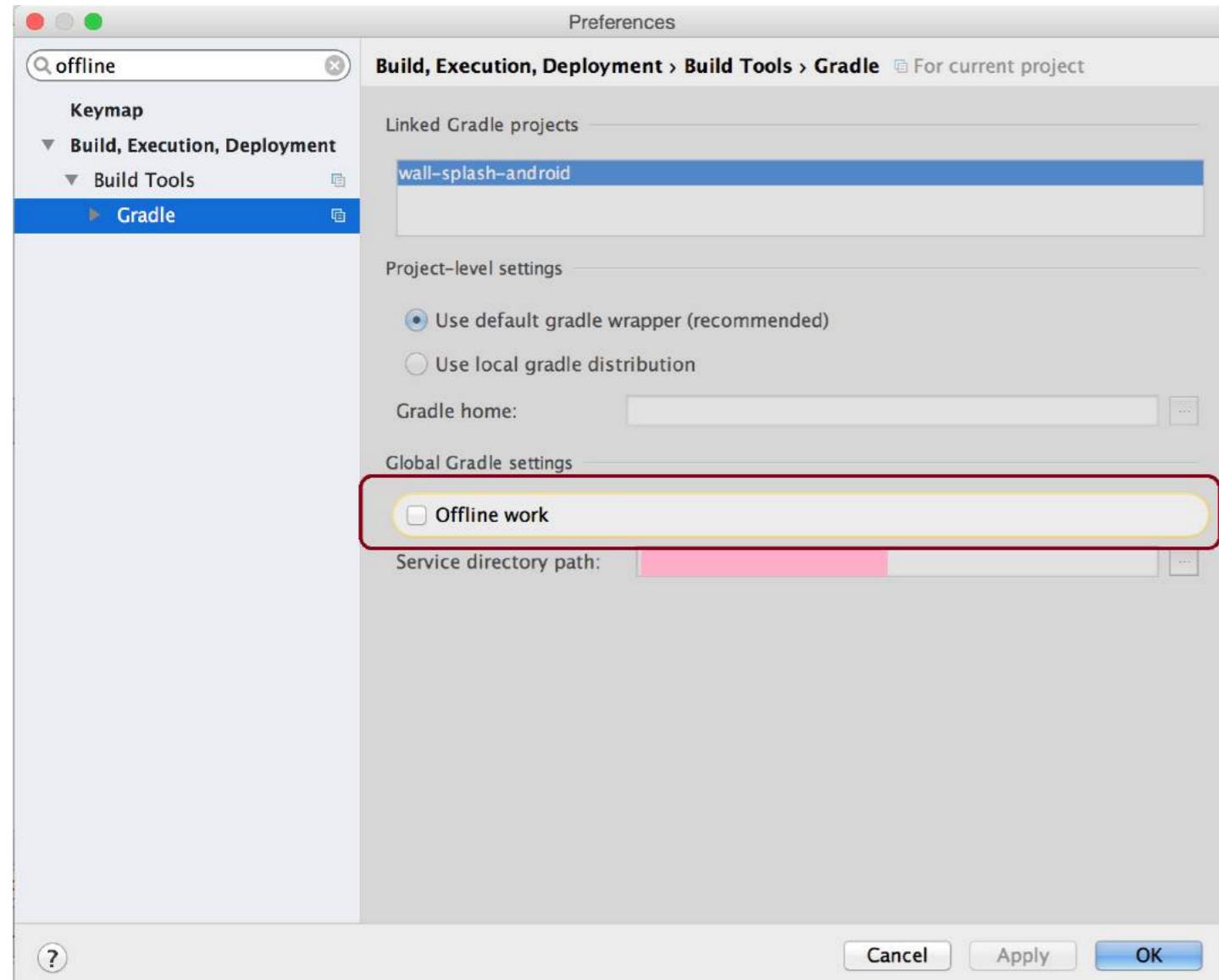
Linux

```
~/.{FOLDER_NAME}/studio.vmoptions and/or ~/.{FOLDER_NAME}/studio64.vmoptions
```

第2.5节：Gradle构建项目耗时过长

Android Studio -> 偏好设置 -> Gradle -> 勾选 离线工作 然后重启你的Android Studio。

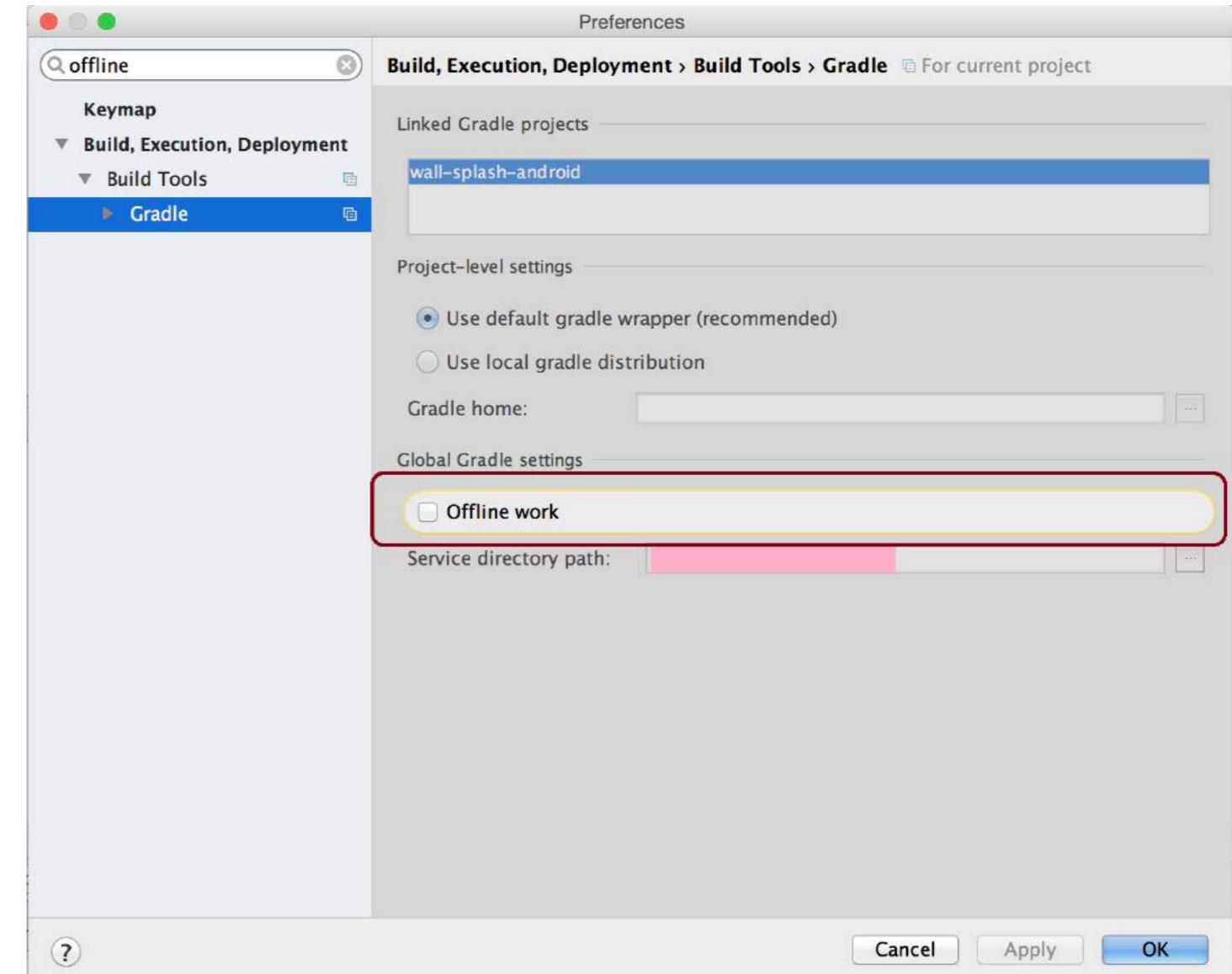
参考截图：



Section 2.5: Gradle build project takes forever

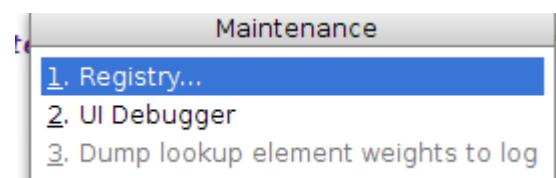
Android Studio -> Preferences -> Gradle -> Tick Offline work and then restart your Android studio.

Reference screenshot:



第2.6节：启用/禁用空行复制

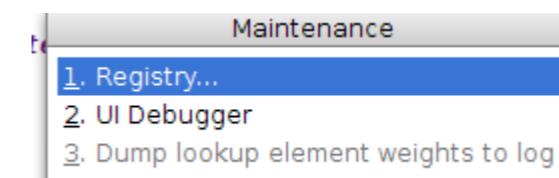
ctrl + alt + shift + / (MacOS上为cmd + alt + shift + /) 应显示以下对话框：



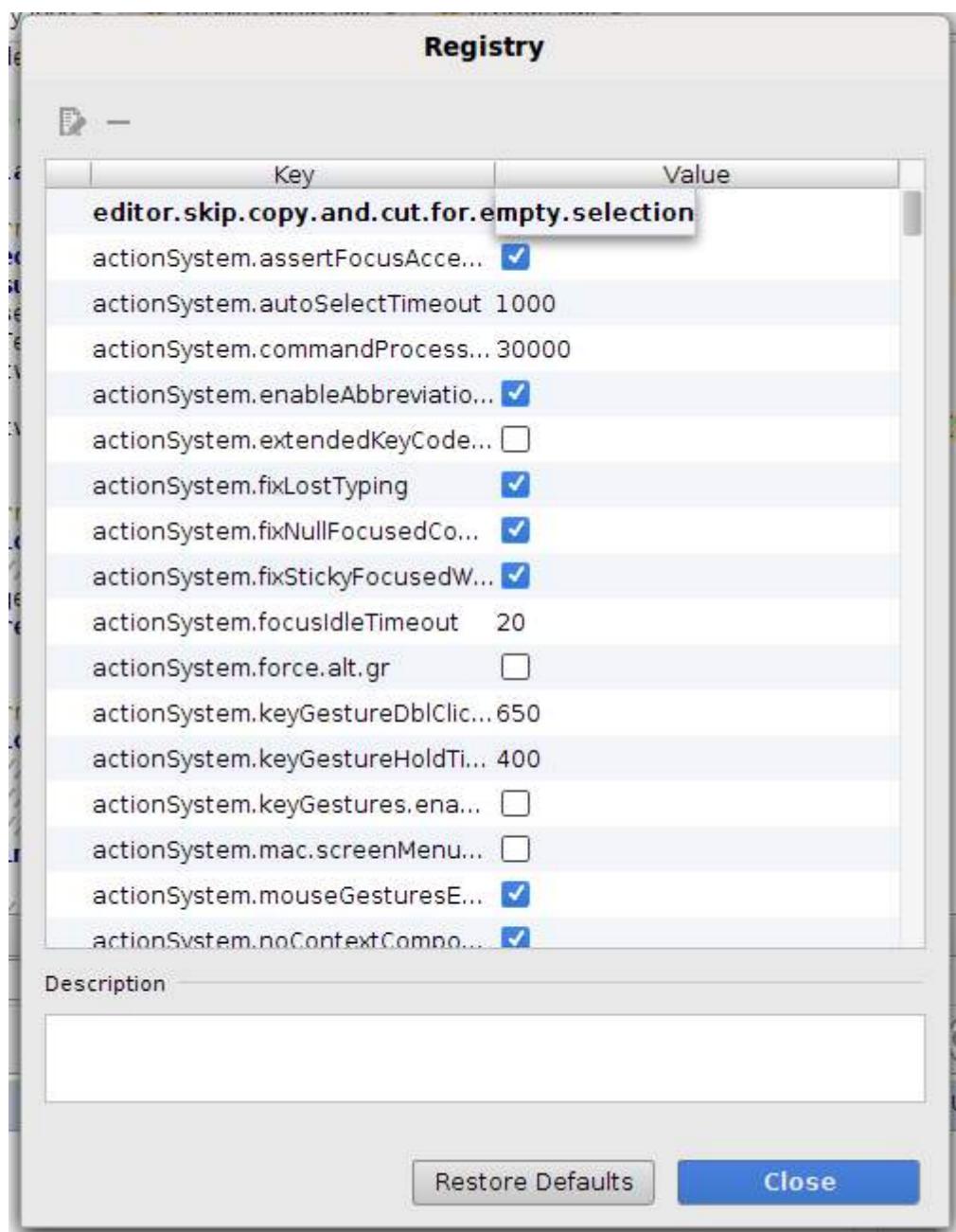
点击注册表你将看到

Section 2.6: Enable/Disable blank line copy

ctrl + alt + shift + / (cmd + alt + shift + / on MacOS) should show you the following dialog:



Clicking on Registry you will get



您想启用/禁用的键是

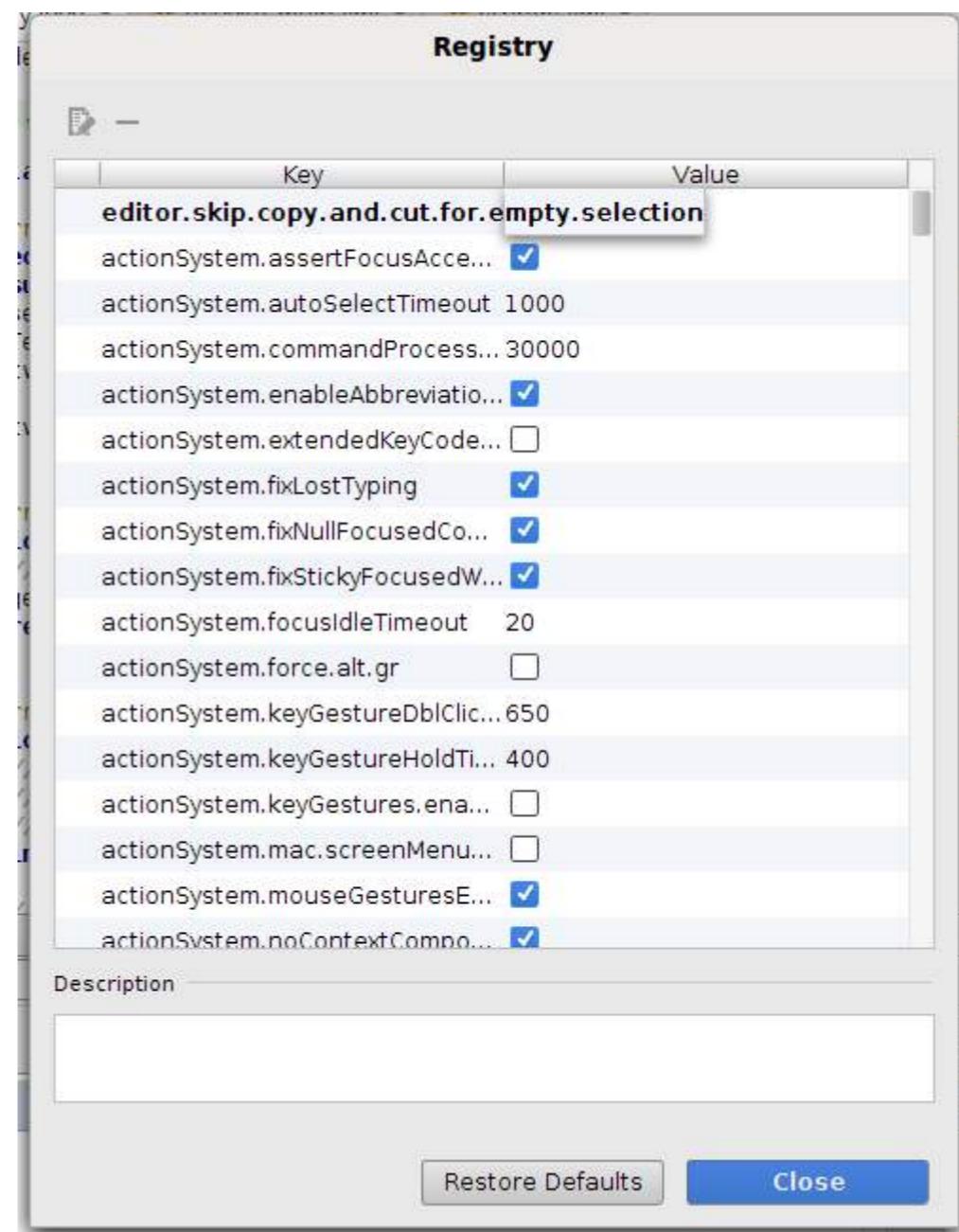
`editor.skip.copy.and.cut.for.empty.selection`

已在Linux Ubuntu和MacOS上测试。

第2.7节：基于消息重要性的logcat消息自定义颜色

进入 文件 -> 设置 -> 编辑器 -> 颜色和字体 -> Android Logcat

根据需要更改颜色：



The key you want to enable/disable is

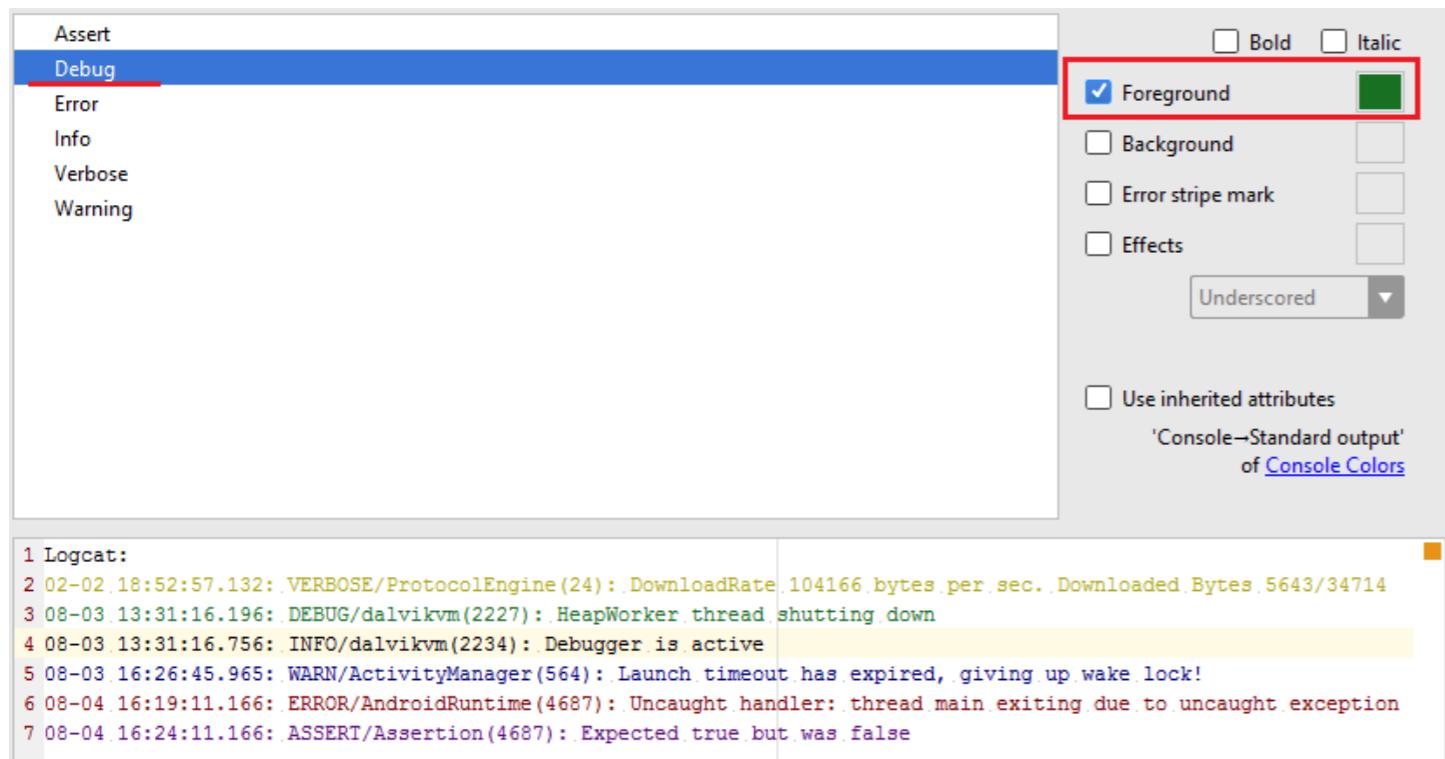
`editor.skip.copy.and.cut.for.empty.selection`

Tested on Linux Ubuntu and MacOS.

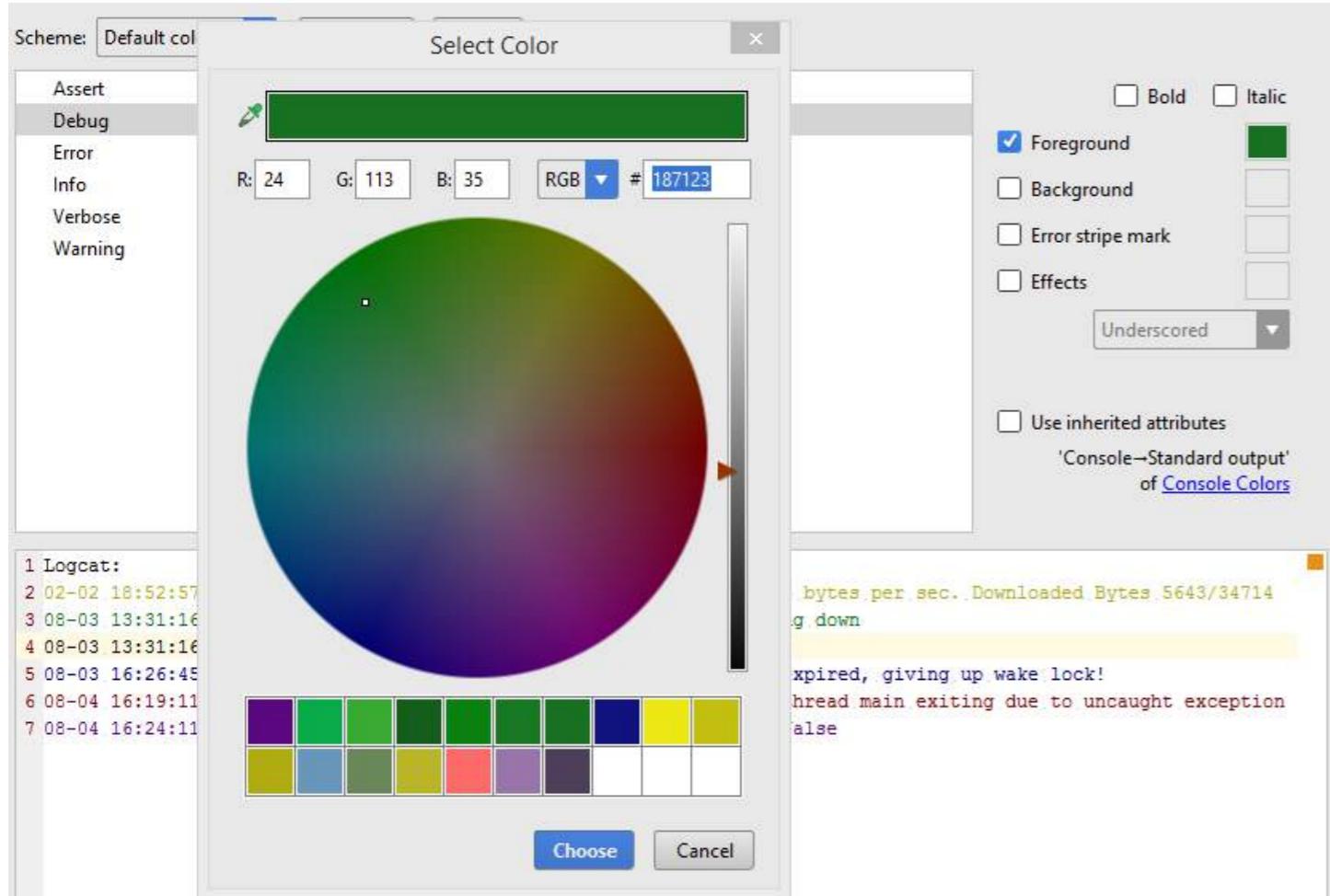
Section 2.7: Custom colors of logcat message based on message importance

Go to File -> Settings -> Editor -> Colors & Fonts -> Android Logcat

Change the colors as you need:

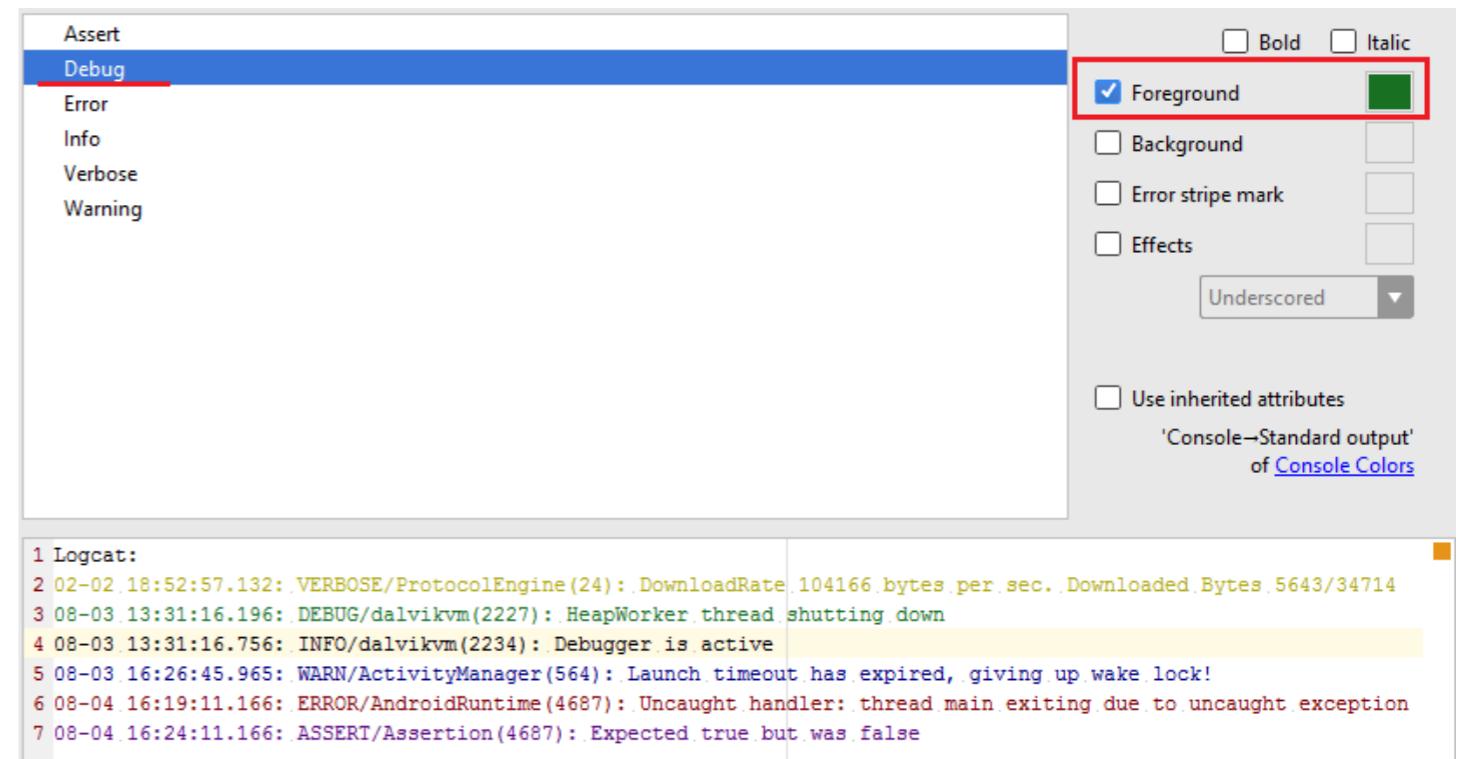


选择合适的颜色：

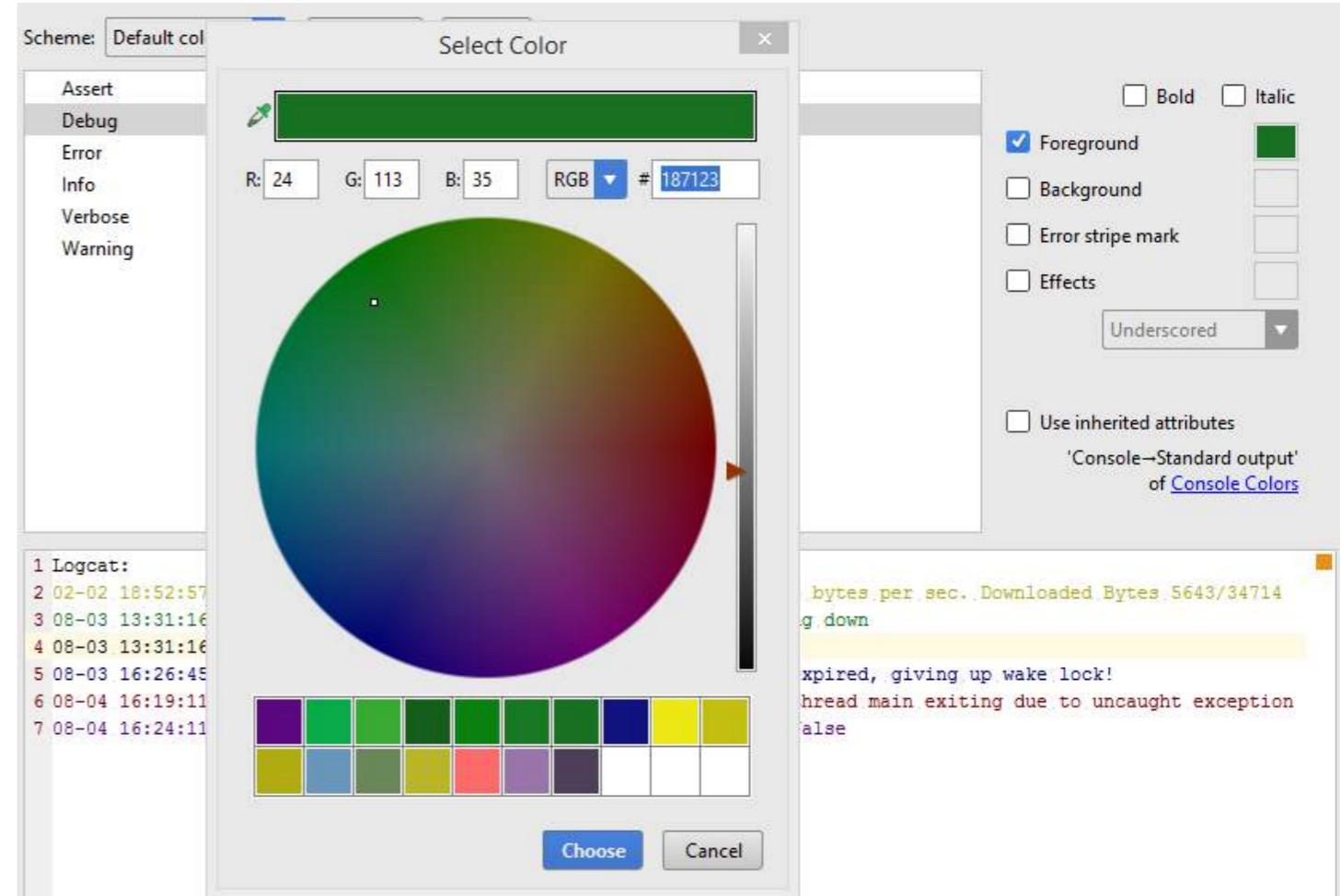


第2.8节：从UI过滤日志

Android日志可以直接从UI中过滤。使用以下代码



Choose the appropriate color:



Section 2.8: Filter logs from UI

Android logs can be filtered directly from the UI. Using this code

```

public class MainActivity extends AppCompatActivity {
    private final static String TAG1 = MainActivity.class.getSimpleName();
    private final static String TAG2 = MainActivity.class.getCanonicalName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.e(TAG1, "Log from onCreate method with TAG1");
        Log.i(TAG2, "Log from onCreate method with TAG2");
    }
}

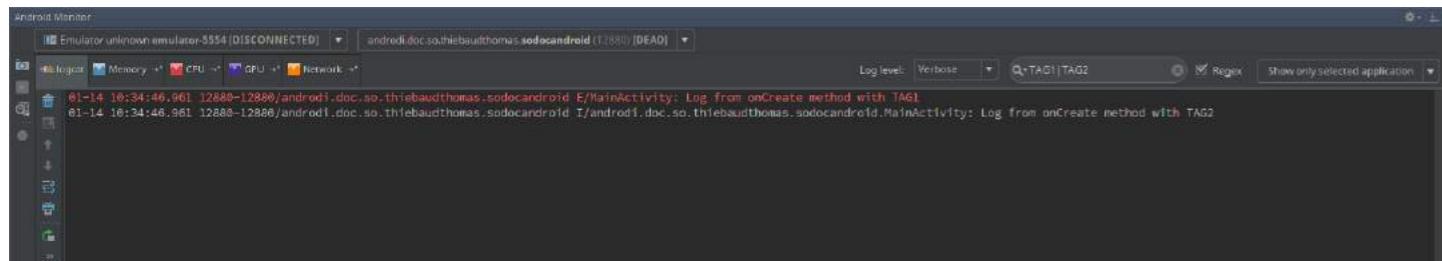
```

如果我使用正则表达式 TAG1|TAG2 和级别 verbose , 我会得到

```

01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: 来自
onCreate 方法的 TAG1 日志
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: 来自 onCreate 方法的 TAG2 日志

```



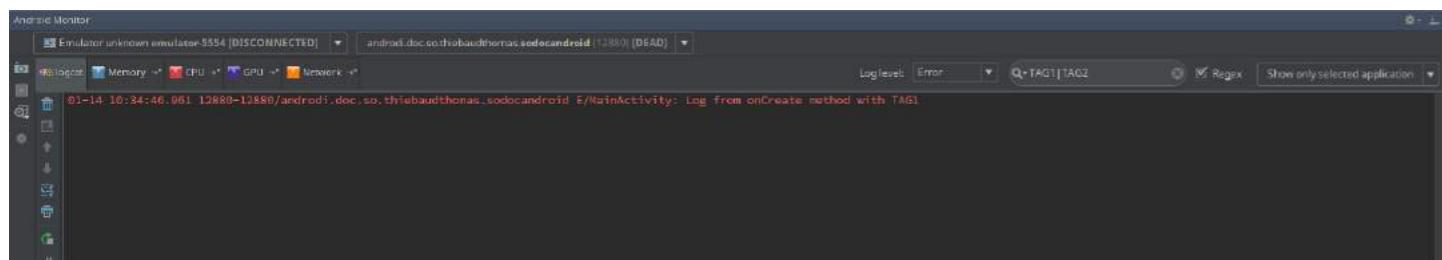
级别可以设置为获取指定级别及以上的日志。例如，verbose 级别将捕获 verbose, debug, info, warn, error 和 assert 日志。

使用相同的例子，如果我将级别设置为 error , 我只会得到

```

01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: 来自
onCreate 方法的 TAG1 日志

```



第2.9节：创建过滤器配置

可以从界面设置并保存自定义过滤器。在 AndroidMonitor 标签页，点击右侧下拉菜单（必须包含 仅显示选定应用 或 无过滤器），然后选择 编辑过滤器配置。

输入你想要的过滤器

```

public class MainActivity extends AppCompatActivity {
    private final static String TAG1 = MainActivity.class.getSimpleName();
    private final static String TAG2 = MainActivity.class.getCanonicalName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.e(TAG1, "Log from onCreate method with TAG1");
        Log.i(TAG2, "Log from onCreate method with TAG2");
    }
}

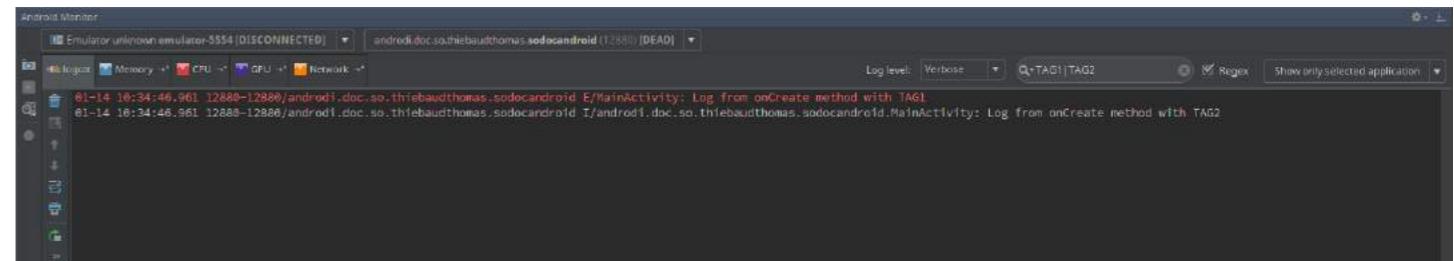
```

If I use the regex TAG1|TAG2 and the level verbose I get

```

01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from
onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2

```



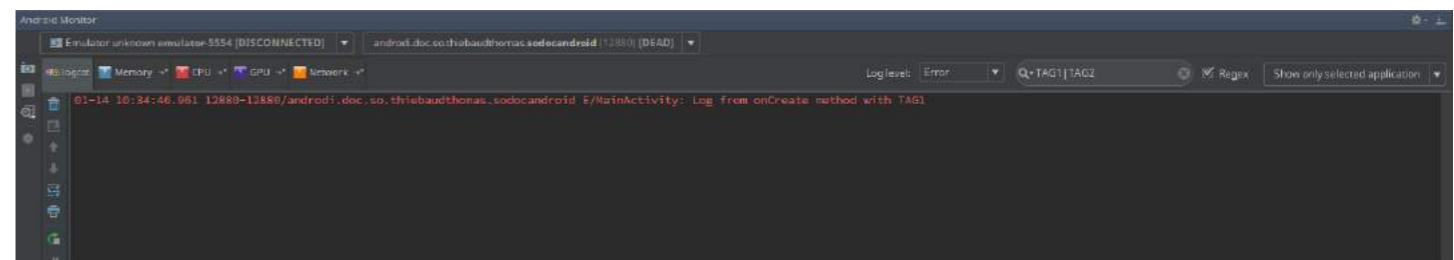
The level can be set to get logs with given level and above. For example the verbose level will catch verbose, debug, info, warn, error and assert logs.

Using the same example, if I set the level to error, I only get

```

01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from
onCreate method with TAG1

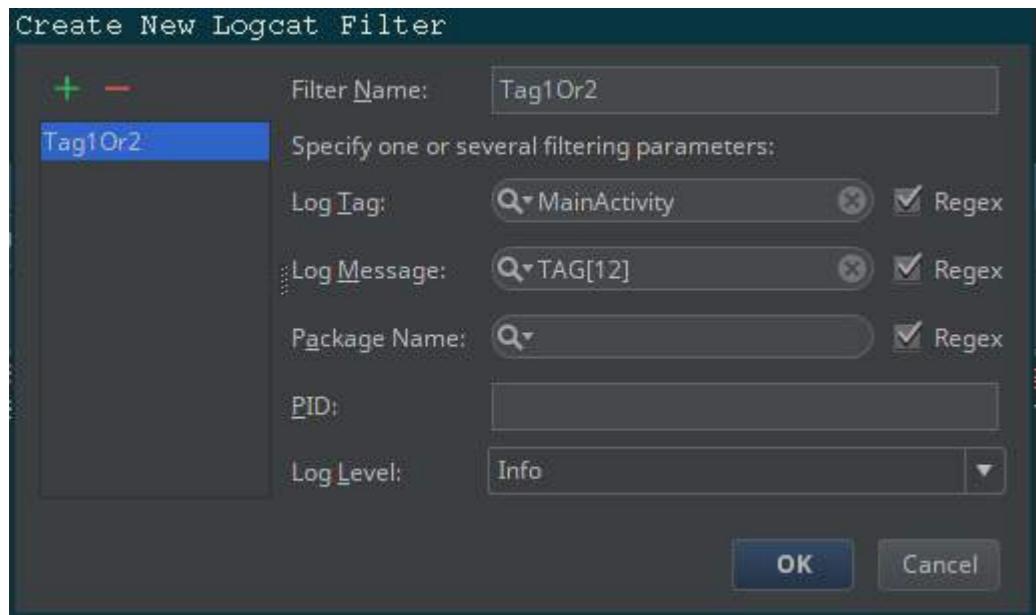
```



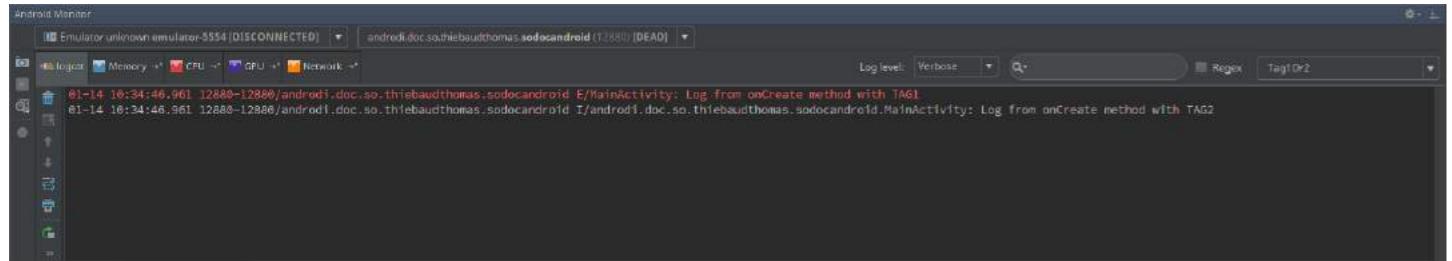
Section 2.9: Create filters configuration

Custom filters can be set and save from the UI. In the AndroidMonitor tab, click on the right dropdown (must contains Show only selected application or No filters) and select Edit filter configuration.

Enter the filter you want

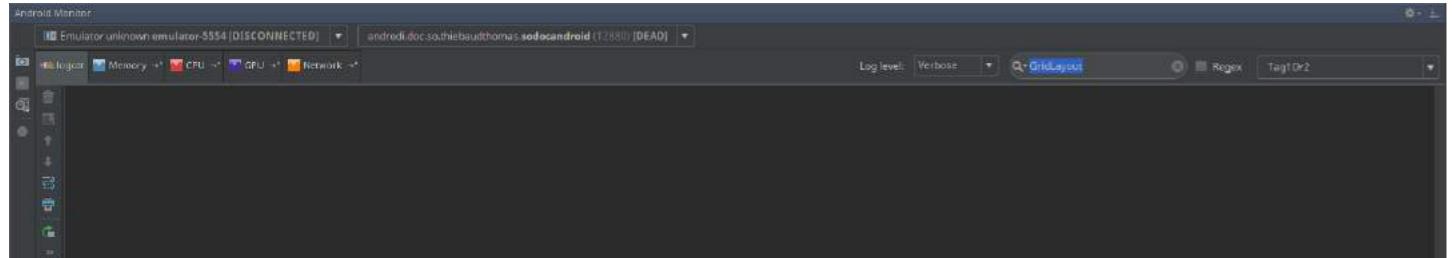


并使用它（你可以从同一个下拉菜单中选择）

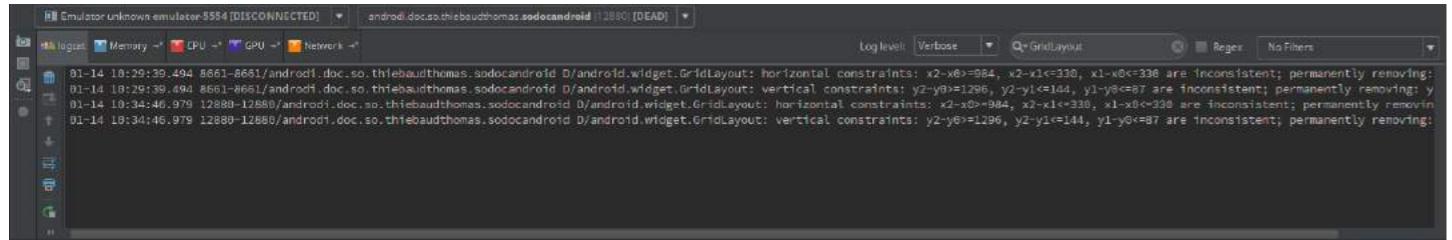


重要 如果你在过滤栏中添加输入，Android Studio 会同时考虑你的过滤条件和输入内容。

当同时有输入和过滤时，没有输出

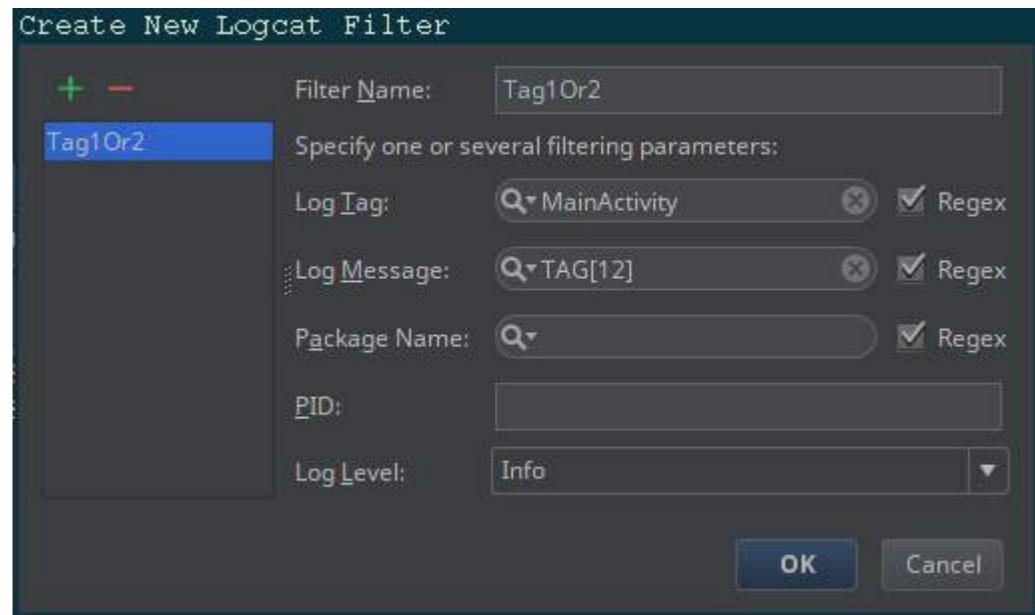


没有过滤时，会有一些输出

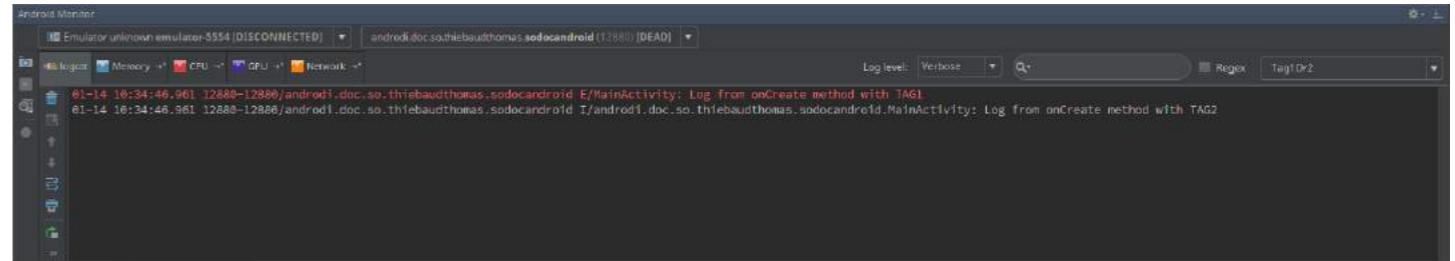


第2.10节：创建assets文件夹

- 在MAIN文件夹上右键 > 新建 > 文件夹 > Assets文件夹。
- Assets文件夹将位于MAIN文件夹下，图标与RES文件夹相同。
- 在此示例中，我放置了一个字体文件。

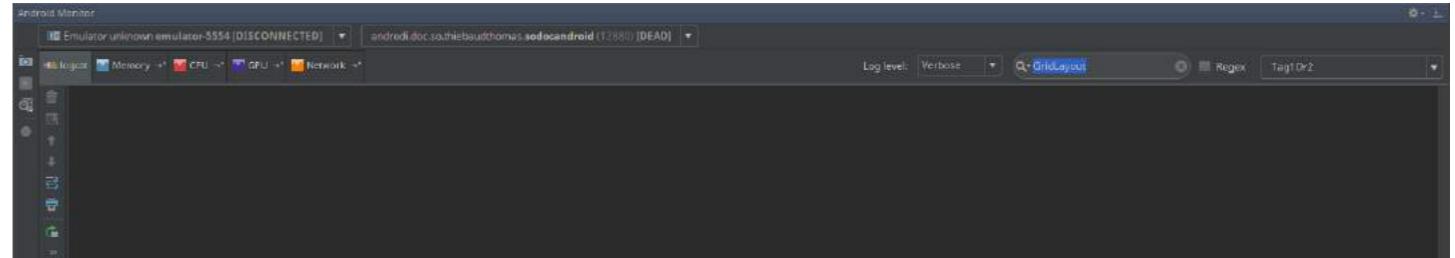


And use it (you can selected it from the same dropdown)

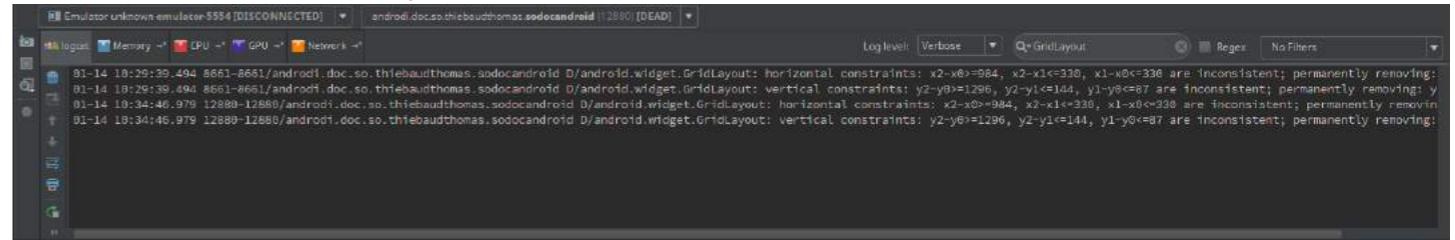


Important If you add an input in the filter bar, android studio will consider both your filter and your input.

With both input and filter there is no output

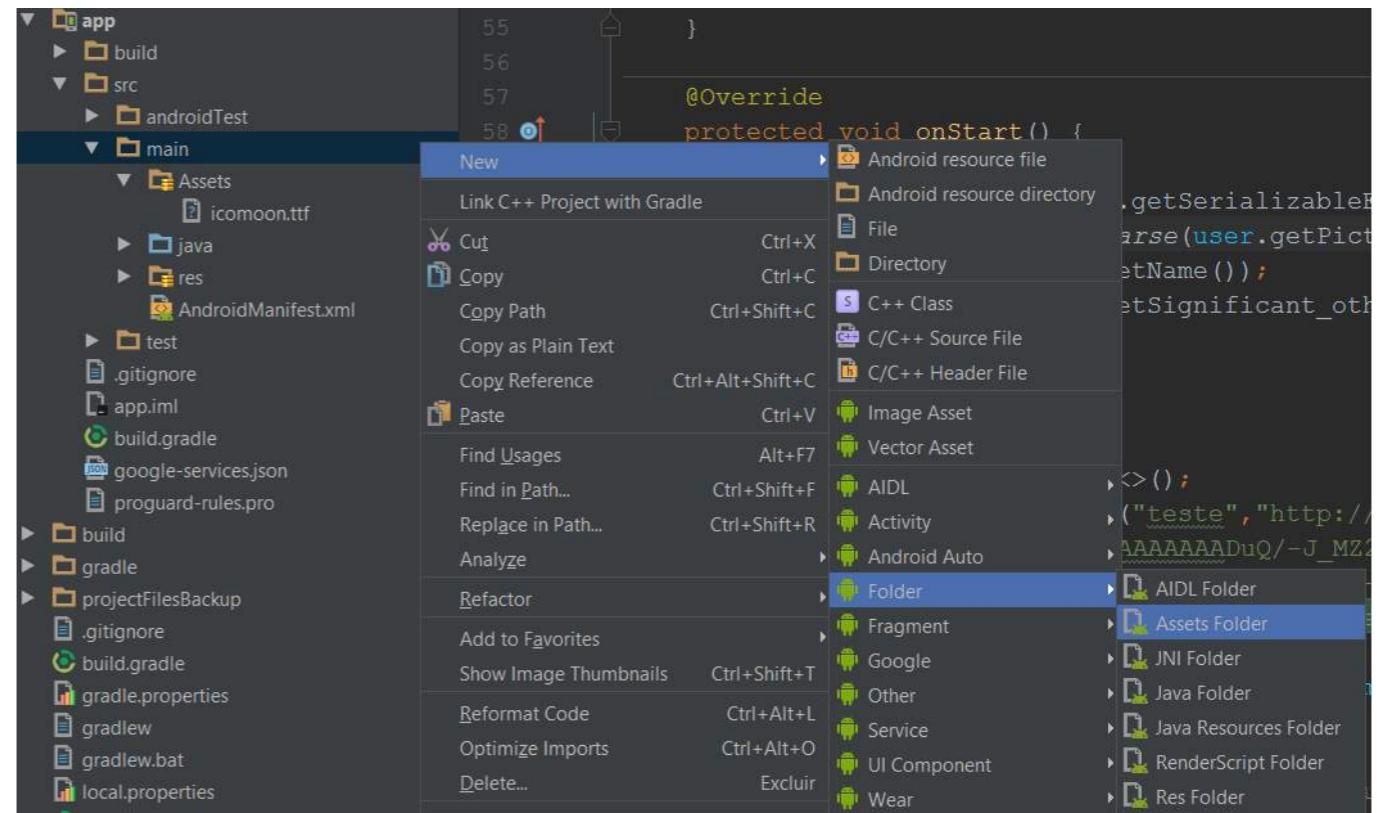
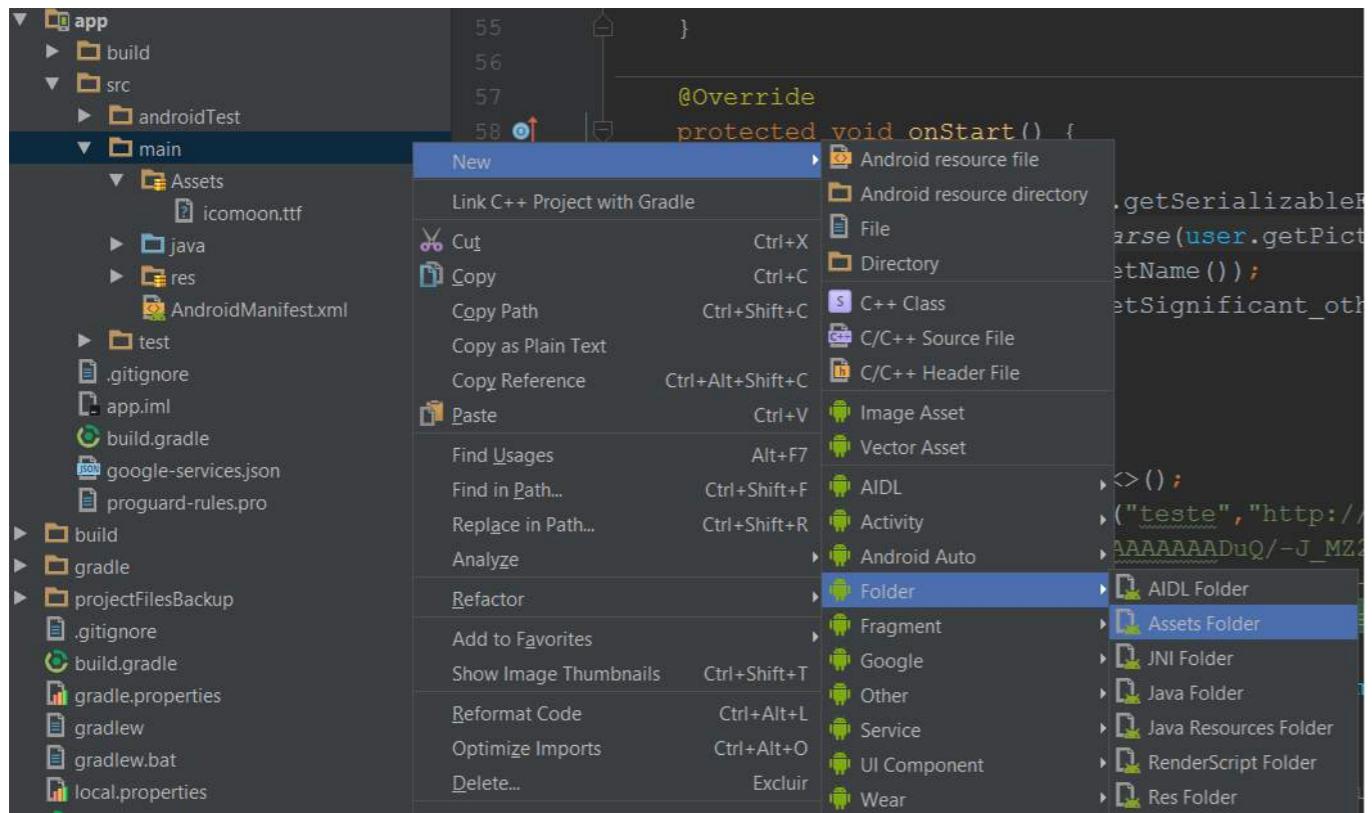


Without filter, there is some outputs



Section 2.10: Create assets folder

- Right click in MAIN folder > New > Folder > Assets Folder.
- Assets folder will be under MAIN folder with the same symbol as RES folder.
- In this example I put a font file.



第3章：Android Studio中的即时运行

第3.1节：启用或禁用即时运行

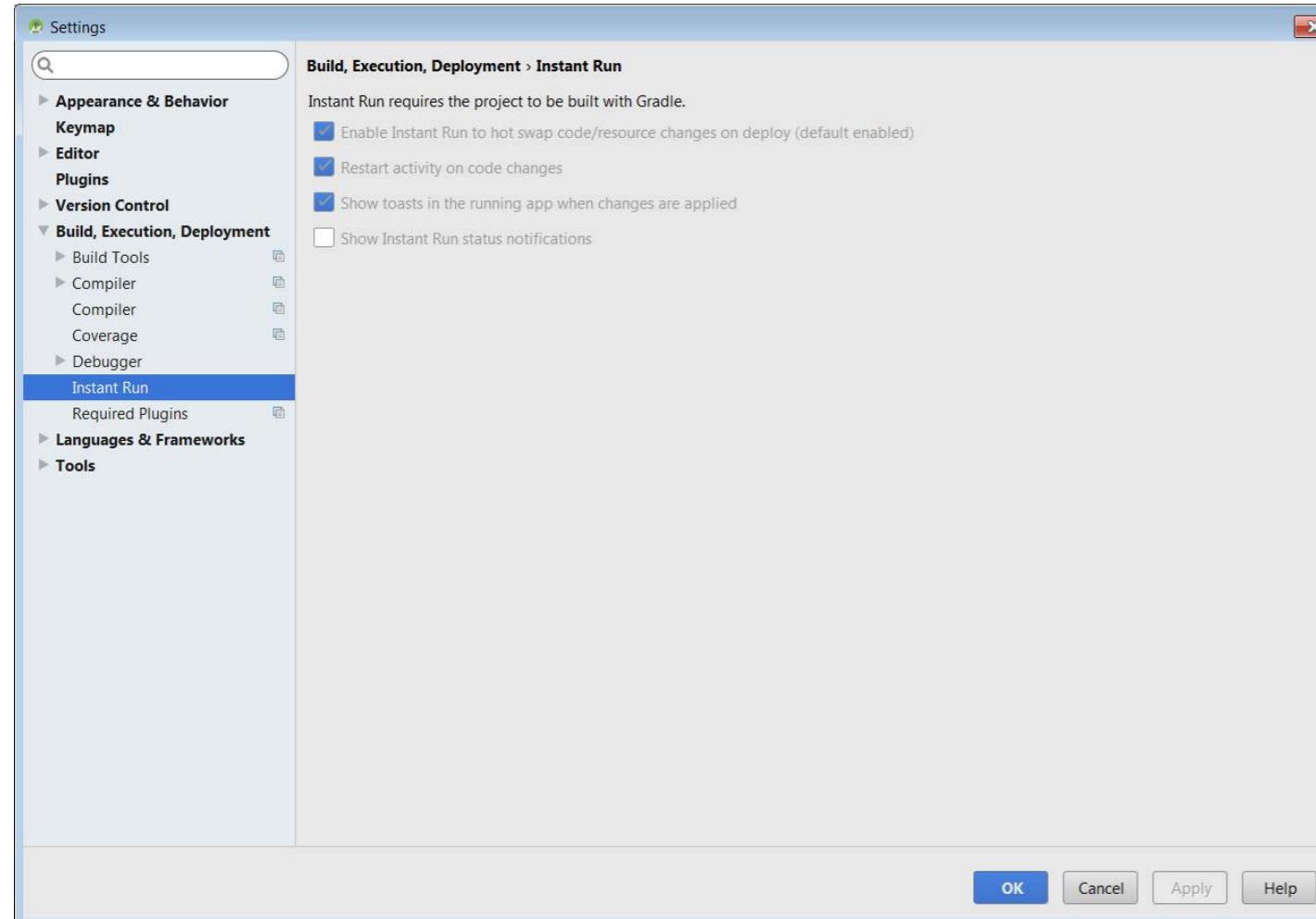
1. 打开设置或偏好设置对话框：

- 在 Windows 或 Linux 上，从主菜单选择文件 > 设置。
- 在 Mac OSX 上，从主菜单选择Android Studio > 偏好设置。

2. 导航到构建、执行、部署 > 编译器。

3. 在命令行选项旁的文本框中输入你的命令行选项。

4. 点击确定以保存并退出。



顶部选项是即时运行。勾选或取消勾选该框。

[文档](#)

第3.2节：即时运行中的代码替换类型

即时运行支持三种代码替换类型，以加快在 Android Studio 中调试和运行应用程序的速度。

- 热替换
- 温替换
- 冷替换

这些交换操作分别在什么时候触发？

Chapter 3: Instant Run in Android Studio

Section 3.1: Enabling or disabling Instant Run

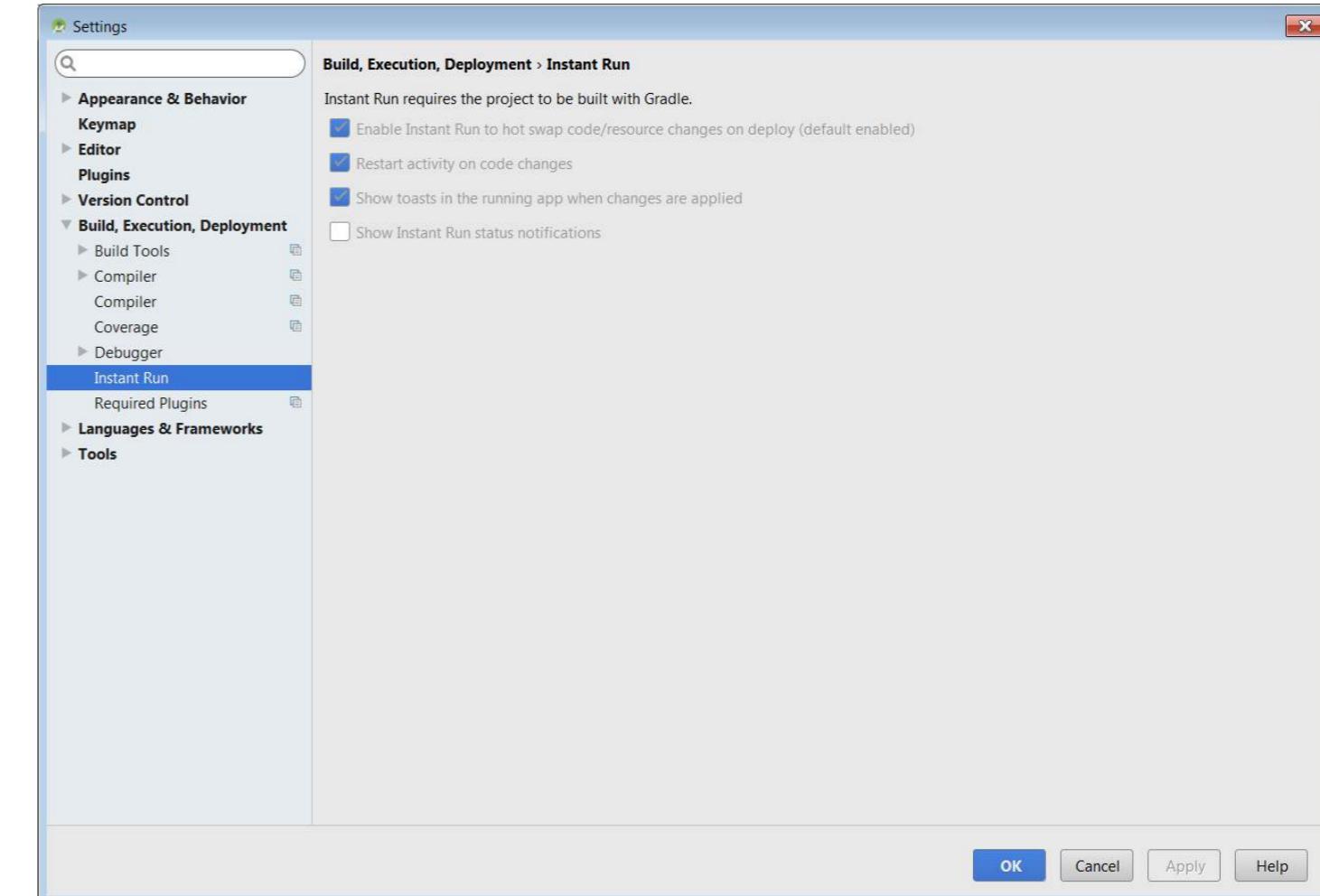
1. Open the Settings or Preferences dialog:

- On Windows or Linux, select **File** > **Settings** from the main menu.
- On Mac OSX, select **Android Studio** > **Preferences** from the main menu.

2. Navigate to Build, Execution, Deployment > Compiler.

3. In the text field next to Command-line Options, enter your command-line options.

4. Click OK to save and exit.



The top option is Instant run. Check/uncheck that box.

[Documentation](#)

Section 3.2: Types of code Swaps in Instant Run

There are three types of code swaps that Instant run enables to support faster debugging and running app from your code in Android Studio.

- Hot Swap
- Warm Swap
- Cold Swap

When are each of these swaps triggered?

热交换 (HOT SWAP) 在现有方法的实现被更改时触发。

温交换 (WARM SWAP) 在现有资源被更改或移除时触发 (res 文件夹中的任何内容)。

冷交换 (COLD SWAP) 每当应用代码中有结构性代码更改时触发，例如：

1. 添加、移除或更改：

- 注解
- 实例字段
- 静态字段
- 静态方法签名
- 实例方法签名

2. 更改当前类继承的父类

3. 更改实现的接口列表

4. 更改类的静态初始化器

5. 重新排序使用动态资源ID的布局元素

代码替换时会发生什么？

热替换 (HOT SWAP) 更改会立即生效——只要下一次调用被更改实现的方法时。

温替换 (WARM SWAP) 重新启动当前活动

冷替换 (COLD SWAP) 重新启动整个应用 (不重新安装)

第3.3节：使用Instant Run时不支持的代码更改

有一些更改Instant Run无法处理，应用将进行完整构建和重新安装，就像Instant Run出现之前一样。

1. 更改应用清单文件
2. 更改应用清单文件引用的资源
3. 更改Android小部件UI元素 (需要清理并重新运行)

[文档](#)

HOT SWAP is triggered when an existing method's implementation is changed.

WARM SWAP is triggered when an existing resource is changed or removed (anything in the res folder)

COLD SWAP whenever there is a structural code change in your app's code e.g.

1. Add, remove, or change:

- an annotation
- an instance field
- a static field
- a static method signature
- an instance method signature

2. Change which parent class the current class inherits from

3. Change the list of implemented interfaces

4. Change a class's static initializer

5. Reorder layout elements that use dynamic resource IDs

What happens when a code swap happens?

HOT SWAP changes are visible instantly - as soon as the next call to the method whose implementation is changed is made.

WARM SWAP restarts the current activity

COLD SWAP restarts the entire app (without reinstall)

Section 3.3: Unsupported code changes when using Instant Run

There are a few changes where instant won't do its trick and a full build and reinstall for your app will happen just like it used to happen before Instant Run was born.

1. Change the app manifest
2. Change resources referenced by the app manifest
3. Change an Android widget UI element (requires a Clean and Rerun)

[Documentation](#)

第4章：TextView

Android SDK中与TextView自定义相关的所有内容

第4.1节：可变文本视图 (Spannable TextView)

在Android中，可以使用可变的TextView来突出显示文本的特定部分，使用不同的颜色、样式、大小和/或点击事件，全部在一个TextView控件中实现。

假设你已经定义了一个TextView，如下所示：

```
TextView textview=findViewById(R.id.textview);
```

然后你可以像下面这样对其应用不同的高亮效果：

- **可变颜色**：为了给文本的某部分设置不同的颜色，可以使用ForegroundSpan，如下例所示：

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new ForegroundColorSpan(firstWordColor), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan(new ForegroundColorSpan(lastWordColor), firstWord.length(),
firstWord.length()+lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textview.setText(spannable);
```

上述代码生成的输出：

Booked
2 rentals

- **可变字体**：为了给文本的某部分设置不同的字体大小，可以使用RelativeSizeSpan，如下例所示：

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new RelativeSizeSpan(1.1f), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // 设置大小
spannable.setSpan(new RelativeSizeSpan(0.8f), firstWord.length(), firstWord.length() +
lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // 设置大小
textview.setText(spannable);
```

上述代码生成的输出：

15
Jun

- **Spannable 字体样式**：为了给部分文本设置不同的字体样式，可以使用自定义的TypefaceSpan，如以下示例所示：

```
Spannable spannable = new SpannableString(firstWord+lastWord);
```

Chapter 4: TextView

Everything related to TextView customization in Android SDK

Section 4.1: Spannable TextView

A spannable TextView can be used in Android to highlight a particular portion of text with a different color, style, size, and/or click event in a single TextView widget.

Consider that you have defined a TextView as follows:

```
TextView textview=findViewById(R.id.textview);
```

Then you can apply different highlighting to it as shown below:

- **Spannable color**: In order to set a different color to some portion of text, a ForegroundColorSpan can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new ForegroundColorSpan(firstWordColor), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan(new ForegroundColorSpan(lastWordColor), firstWord.length(),
firstWord.length()+lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textview.setText(spannable);
```

Output created by the code above:

Booked
2 rentals

- **Spannable font**: In order to set a different font size to some portion of text, a RelativeSizeSpan can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new RelativeSizeSpan(1.1f), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
spannable.setSpan(new RelativeSizeSpan(0.8f), firstWord.length(), firstWord.length() +
lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
textview.setText(spannable);
```

Output created by the code above:

15
Jun

- **Spannable typeface**: In order to set a different font typeface to some portion of text, a custom TypefaceSpan can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
```

```

spannable.setSpan( new CustomTypefaceSpan("SFUIText-Bold.otf",fontBold), 0,
firstWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Regular.otf",fontRegular),
firstWord.length(), firstWord.length() + lastWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
text.setText( spannable );

```

但是，为了使上述代码生效，类CustomTypefaceSpan必须继承自类TypefaceSpan。实现方式如下：

```

public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String 字体族, Typeface 类型) {
        super(字体族);
        newType = 类型;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint 画笔, Typeface tf) {
        int 旧样式;
        Typeface 旧字体 = 画笔.getTypeface();
        if (旧字体 == null) {
            旧样式 = 0;
        } else {
            旧样式 = 旧字体.getStyle();
        }
        int 伪造 = 旧样式 & ~tf.getStyle();
        if ((伪造 & Typeface.BOLD) != 0) {
            画笔.setFakeBoldText(true);
        }

        if ((伪造 & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }
        paint.setTypeface(tf);
    }
}

```

```

spannable.setSpan( new CustomTypefaceSpan("SFUIText-Bold.otf",fontBold), 0,
firstWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Regular.otf",fontRegular),
firstWord.length(), firstWord.length() + lastWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
text.setText( spannable );

```

However, in order to make the above code working, the class CustomTypefaceSpan has to be derived from the class TypefaceSpan. This can be done as follows:

```

public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String family, Typeface type) {
        super(family);
        newType = type;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint paint, Typeface tf) {
        int oldStyle;
        Typeface old = paint.getTypeface();
        if (old == null) {
            oldStyle = 0;
        } else {
            oldStyle = old.getStyle();
        }
        int fake = oldStyle & ~tf.getStyle();
        if ((fake & Typeface.BOLD) != 0) {
            paint.setFakeBoldText(true);
        }

        if ((fake & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }
        paint.setTypeface(tf);
    }
}

```

第4.2节：删除线文本视图

对整个文本添加删除线

```

String sampleText = "这是一个测试删除线";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);

```

输出： 这是一个测试删除线

Section 4.2: Strikethrough TextView

Strikethrough the entire text

```

String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);

```

Output: This is a test strike

仅对部分文本添加删除线

```
String sampleText = "这是一个测试删除线";
SpannableStringBuilder spanBuilder = new SpannableStringBuilder(sampleText);
StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spanBuilder.setSpan(
    strikethroughSpan, // 要添加的Span
    0, // 起始位置
    4, // 跨度结束 (不包括)
    SPAN_EXCLUSIVE_EXCLUSIVE // 文本更改不会反映在删除线更改中
);
textView.setText(spanBuilder);
```

输出： 这是一个测试删除线

第4.3节：带图片的TextView

Android允许程序员在TextView的四个角落放置图片。例如，如果你正在创建一个带有TextView的字段，同时你想显示该字段是可编辑的，那么开发者通常会在该字段附近放置一个编辑图标。Android为TextView提供了一个有趣的选项，称为compound drawable：

```
<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:drawablePadding="4dp"
    android:drawableRight="@drawable/edit"
    android:text="Hello world"
    android:textSize="18dp" />
```

你可以将drawable设置到TextView的任意一侧，方法如下：

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

设置可绘制对象也可以通过以下编程方式实现：

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

将传递给setCompoundDrawables()的任何参数设置为null将移除TextView对应边的图标。

第4.4节：使RelativeSizeSpan顶部对齐

为了使RelativeSizeSpan顶部对齐，可以从类SuperscriptSpan派生一个自定义类。以下示例中，派生类命名为TopAlignSuperscriptSpan：

activity_main.xml：

```
<TextView
    android:id="@+id/txtView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
```

Strikethrough only parts of the text

```
String sampleText = "This is a test strike";
SpannableStringBuilder spanBuilder = new SpannableStringBuilder(sampleText);
StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spanBuilder.setSpan(
    strikethroughSpan, // Span to add
    0, // Start
    4, // End of the span (exclusive)
    SPAN_EXCLUSIVE_EXCLUSIVE // Text changes will not reflect in the strike changing
);
textView.setText(spanBuilder);
```

Output: This is a test strike

Section 4.3: TextView with image

Android allows programmers to place images at all four corners of a TextView. For example, if you are creating a field with a TextView and at same time you want to show that the field is editable, then developers will usually place an edit icon near that field. Android provides us an interesting option called **compound drawable** for a TextView:

```
<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:drawablePadding="4dp"
    android:drawableRight="@drawable/edit"
    android:text="Hello world"
    android:textSize="18dp" />
```

You can set the drawable to any side of your TextView as follows:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

Setting the drawable can also be achieved programmatically in the following way:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

Setting any of the parameters handed over to setCompoundDrawables() to **null** will remove the icon from the corresponding side of the TextView.

Section 4.4: Make RelativeSizeSpan align to top

In order to make a RelativeSizeSpan align to the top, a custom class can be derived from the class SuperscriptSpan. In the following example, the derived class is named TopAlignSuperscriptSpan:

activity_main.xml:

```
<TextView
    android:id="@+id/txtView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
```

```
        android:textSize="26sp" />
```

MainActivity.java :

```
TextView txtView = (TextView) findViewById(R.id.txtView);

SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java :

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //将上标除以此数字
    protected int fontScale = 2;

    //偏移值, 范围0到1.0
    protected float shiftPercentage = 0;

    //不偏移
    TopAlignSuperscriptSpan() {}

    //设置偏移百分比
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

    @Override
    public void updateDrawState( TextPaint tp ) {
        //原始 ascent
        float ascent = tp.ascent();

        //缩小字体
        tp.setTextSize( tp.getTextSize() / fontScale );

        //获取新的字体上升高度
        float newAscent = tp.getFontMetrics().ascent;

        //将基线移动到旧字体顶部, 然后向下移动新字体的大小
        //通过偏移百分比调整误差
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
            - ( newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

```
        android:textSize="26sp" />
```

MainActivity.java:

```
TextView txtView = (TextView) findViewById(R.id.txtView);

SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java:

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //divide superscript by this number
    protected int fontScale = 2;

    //shift value, 0 to 1.0
    protected float shiftPercentage = 0;

    //doesn't shift
    TopAlignSuperscriptSpan() {}

    //sets the shift percentage
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

    @Override
    public void updateDrawState( TextPaint tp ) {
        //original ascent
        float ascent = tp.ascent();

        //scale down the font
        tp.setTextSize( tp.getTextSize() / fontScale );

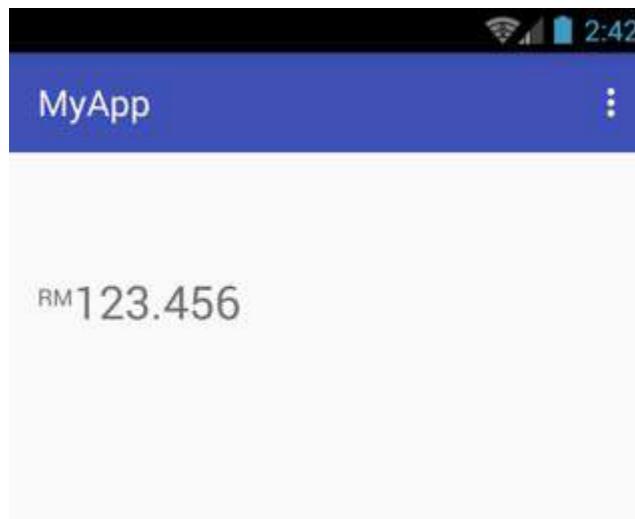
        //get the new font ascent
        float newAscent = tp.getFontMetrics().ascent;

        //move baseline to top of old font, then move down size of new font
        //adjust for errors with shift percentage
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
            - ( newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

参考截图：

Reference screenshot:



第4.5节：TextView上的捏合缩放

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="这是我的捏合缩放演示示例文本，您可以使用捏合缩放进行放大和缩小，谢谢" />

</RelativeLayout>
```

MainActivity.java:

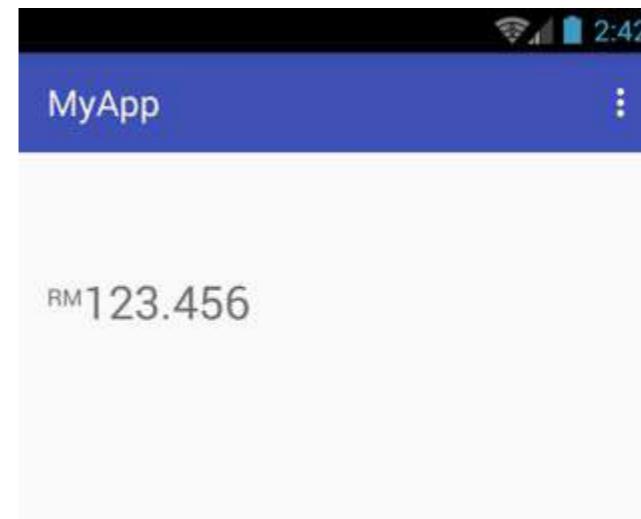
```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mtv);
```



Section 4.5: Pinchzoom on TextView

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="This is my sample text for pinch zoom demo, you can zoom in and out using
pinch zoom, thanks" />

</RelativeLayout>
```

MainActivity.java:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mtv);
```

```

mytv.setTextSize(mRatio + 13);
}

public boolean onTouchEvent(MotionEvent event) {
    if (event.getPointerCount() == 2) {
        int action = event.getAction();
        int pureaction = action & MotionEvent.ACTION_MASK;
        if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
            mBaseDist = getDistance(event);
            mBaseRatio = mRatio;
        } 否则 {
            float delta = (getDistance(event) - mBaseDist) / STEP;
            float multi = (float) Math.pow(2, delta);
            mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
            mytv.setTextSize(mRatio + 13);
        }
    }
    return true;
}

int getDistance(MotionEvent event) {
    int dx = (int) (event.getX(0) - event.getX(1));
    int dy = (int) (event.getY(0) - event.getY(1));
    return (int) (Math.sqrt(dx * dx + dy * dy));
}

public boolean onTouch(View v, MotionEvent event) {
    return false;
}
}

```

第4.6节：具有不同文本大小的TextView

您可以通过Span在TextView中实现不同的文本大小

```

TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)

```

第4.7节：主题和样式定制

MainActivity.java :

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

activity_main.xml :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"

```

```

mytv.setTextSize(mRatio + 13);
}

public boolean onTouchEvent(MotionEvent event) {
    if (event.getPointerCount() == 2) {
        int action = event.getAction();
        int pureaction = action & MotionEvent.ACTION_MASK;
        if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
            mBaseDist = getDistance(event);
            mBaseRatio = mRatio;
        } else {
            float delta = (getDistance(event) - mBaseDist) / STEP;
            float multi = (float) Math.pow(2, delta);
            mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
            mytv.setTextSize(mRatio + 13);
        }
    }
    return true;
}

int getDistance(MotionEvent event) {
    int dx = (int) (event.getX(0) - event.getX(1));
    int dy = (int) (event.getY(0) - event.getY(1));
    return (int) (Math.sqrt(dx * dx + dy * dy));
}

public boolean onTouch(View v, MotionEvent event) {
    return false;
}
}

```

Section 4.6: TextView with different Textsize

You can archive different Textsizes inside a TextView with a Span

```

TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)

```

Section 4.7: Theme and Style customization

MainActivity.java:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

activity_main.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        tools:context=".MainActivity">

    <com.customthemattributedemo.customview.CustomTextView
        style="?mediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />

    <com.customthemattributedemo.customview.CustomTextView
        style="?largeTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />
</LinearLayout>

```

CustomTextView.java :

```

public class CustomTextView extends TextView {

    private static final String TAG = "TextViewPlus";
    private Context mContext;

    public CustomTextView(Context context) {
        super(context);
        mContext = context;
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        setCustomFont(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray customFontNameTypedArray = ctx.obtainStyledAttributes(attrs,
R.styleable.CustomTextView);
        字符串 customFont =
customFontNameTypedArray.getString(R.styleable.CustomTextView_font_family);
        Typeface typeface = null;
        typeface = Typeface.createFromAsset(ctx.getAssets(), customFont);
        setTypeface(typeface);
        customFontNameTypedArray.recycle();
    }
}

```

attrs.xml:

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        tools:context=".MainActivity">

    <com.customthemattributedemo.customview.CustomTextView
        style="?mediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />

    <com.customthemattributedemo.customview.CustomTextView
        style="?largeTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />
</LinearLayout>

```

CustomTextView.java:

```

public class CustomTextView extends TextView {

    private static final String TAG = "TextViewPlus";
    private Context mContext;

    public CustomTextView(Context context) {
        super(context);
        mContext = context;
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        setCustomFont(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray customFontNameTypedArray = ctx.obtainStyledAttributes(attrs,
R.styleable.CustomTextView);
        String customFont =
customFontNameTypedArray.getString(R.styleable.CustomTextView_font_family);
        Typeface typeface = null;
        typeface = Typeface.createFromAsset(ctx.getAssets(), customFont);
        setTypeface(typeface);
        customFontNameTypedArray.recycle();
    }
}

```

attrs.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

<attr name="mediumTextStyle" format="reference" />
<attr name="largeTextStyle" format="reference" />

<declare-styleable name="CustomTextView">

    <attr name="font_family" format="string" />
    <!-- 你的其他属性 -->

</declare-styleable>
</resources>

```

strings.xml:

```

<resources>
    <string name="app_name">自定义样式主题属性演示</string>
    <string name="message_hello">你好 Hiren ! </string>

    <string name="bold_font">bold.ttf</string>
</resources>

```

styles.xml :

```

<resources>

<!-- 基础应用主题。 -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- 在此自定义你的主题。 -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>

    <item name="mediumTextStyle">@style/textMedium</item>
    <item name="largeTextStyle">@style/textLarge</item>
</style>

<style name="textMedium" parent="textParentStyle">
    <item name="android:textAppearance">@android:style/TextAppearance.Medium</item>
</style>

<style name="textLarge" parent="textParentStyle">
    <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
</style>

<style name="textParentStyle">
    <item name="android:textColor">@android:color/white</item>
    <item name="android:background">@color/colorPrimary</item>
    <item name="android:padding">5dp</item>
</style>

</resources>

```

第4.8节：TextView自定义

```
public class CustomTextView extends TextView {
```

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

<attr name="mediumTextStyle" format="reference" />
<attr name="largeTextStyle" format="reference" />

<declare-styleable name="CustomTextView">

    <attr name="font_family" format="string" />
    <!-- Your other attributes -->

</declare-styleable>
</resources>

```

strings.xml:

```

<resources>
    <string name="app_name">Custom Style Theme Attribute Demo</string>
    <string name="message_hello">Hello Hiren!</string>

    <string name="bold_font">bold.ttf</string>
</resources>

```

styles.xml:

```

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>

    <item name="mediumTextStyle">@style/textMedium</item>
    <item name="largeTextStyle">@style/textLarge</item>
</style>

<style name="textMedium" parent="textParentStyle">
    <item name="android:textAppearance">@android:style/TextAppearance.Medium</item>
</style>

<style name="textLarge" parent="textParentStyle">
    <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
</style>

<style name="textParentStyle">
    <item name="android:textColor">@android:color/white</item>
    <item name="android:background">@color/colorPrimary</item>
    <item name="android:padding">5dp</item>
</style>

</resources>

```

Section 4.8: TextView customization

```
public class CustomTextView extends TextView {
```

```

private float strokeWidth;
private Integer strokeColor;
private Paint.Join strokeJoin;
private float strokeMiter;

public CustomTextView(Context context) {
    super(context);
    init(null);
}

public CustomTextView(Context context, AttributeSet attrs) {
    super(context, attrs);
    init(attrs);
}

public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    init(attrs);
}

public void init(AttributeSet attrs) {

    if (attrs != null) {
        TypedArray a = getContext().obtainStyledAttributes(attrs, R.styleable.CustomTextView);

        if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
            float strokeWidth = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeWidth,
1);
            int strokeColor = a.getColor(R.styleable.CustomTextView_strokeColor, 0xff000000);
            float strokeMiter = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeMiter,
10);
            Paint.Join strokeJoin = null;
            switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                case (0):
                    strokeJoin = Paint.Join.MITER;
                    break;
                case (1):
                    strokeJoin = Paint.Join.BEVEL;
                    break;
                case (2):
                    strokeJoin = Paint.Join.ROUND;
                    break;
            }
            this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
        }
    }
}

public void setStroke(float width, int color, Paint.Join join, float miter) {
    strokeWidth = width;
    strokeColor = color;
    strokeJoin = join;
    strokeMiter = miter;
}

@Override
public void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    int restoreColor = this.getCurrentTextColor();
    if (strokeColor != null) {

```

```

private float strokeWidth;
private Integer strokeColor;
private Paint.Join strokeJoin;
private float strokeMiter;

public CustomTextView(Context context) {
    super(context);
    init(null);
}

public CustomTextView(Context context, AttributeSet attrs) {
    super(context, attrs);
    init(attrs);
}

public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    init(attrs);
}

public void init(AttributeSet attrs) {

    if (attrs != null) {
        TypedArray a = getContext().obtainStyledAttributes(attrs, R.styleable.CustomTextView);

        if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
            float strokeWidth = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeWidth,
1);
            int strokeColor = a.getColor(R.styleable.CustomTextView_strokeColor, 0xff000000);
            float strokeMiter = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeMiter,
10);
            Paint.Join strokeJoin = null;
            switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                case (0):
                    strokeJoin = Paint.Join.MITER;
                    break;
                case (1):
                    strokeJoin = Paint.Join.BEVEL;
                    break;
                case (2):
                    strokeJoin = Paint.Join.ROUND;
                    break;
            }
            this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
        }
    }
}

public void setStroke(float width, int color, Paint.Join join, float miter) {
    strokeWidth = width;
    strokeColor = color;
    strokeJoin = join;
    strokeMiter = miter;
}

@Override
public void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    int restoreColor = this.getCurrentTextColor();
    if (strokeColor != null) {

```

```

TextPaint paint = this.getPaint();
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeJoin(strokeJoin);
paint.setStrokeMiter(strokeMiter);
this.setTextColor(strokeColor);
paint.setStrokeWidth(strokeWidth);
super.onDraw(canvas);
paint.setStyle(Paint.Style.FILL);
this.setTextColor	restoreColor;
}
}
}

```

用法：

```

public class MainActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

CustomTextView customTextView = (CustomTextView) findViewById(R.id.pager_title);
}
}

```

布局：

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">

<pk.sohail.gallerytest.activity.CustomTextView
    android:id="@+id/pager_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:gravity="center"
    android:text="@string/txt_title_photo_gallery"
    android:textColor="@color/white"
    android:textSize="30dp"
    android:textStyle="bold"
    app:outerShadowRadius="10dp"
    app:strokeColor="@color/title_text_color"
    app:strokeJoinStyle="miter"
    app:strokeWidth="2dp" />

</RelativeLayout>

```

属性：

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

<declare-styleable name="CustomTextView">

```

```

TextPaint paint = this.getPaint();
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeJoin(strokeJoin);
paint.setStrokeMiter(strokeMiter);
this.setTextColor(strokeColor);
paint.setStrokeWidth(strokeWidth);
super.onDraw(canvas);
paint.setStyle(Paint.Style.FILL);
this.setTextColor(restoreColor);
}
}
}

```

Usage:

```

public class MainActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

CustomTextView customTextView = (CustomTextView) findViewById(R.id.pager_title);
}
}

```

Layout:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">

<pk.sohail.gallerytest.activity.CustomTextView
    android:id="@+id/pager_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:gravity="center"
    android:text="@string/txt_title_photo_gallery"
    android:textColor="@color/white"
    android:textSize="30dp"
    android:textStyle="bold"
    app:outerShadowRadius="10dp"
    app:strokeColor="@color/title_text_color"
    app:strokeJoinStyle="miter"
    app:strokeWidth="2dp" />

</RelativeLayout>

```

attars:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

<declare-styleable name="CustomTextView">

```

```

<attr name="outerShadowRadius" format="dimension" />
<attr name="strokeWidth" format="dimension" />
<attr name="strokeMiter" format="dimension" />
<attr name="strokeColor" format="color" />
<attr name="strokeJoinStyle">
    <enum name="斜接" value="0" />
    <enum name="斜角" value="1" />
    <enum name="圆角" value="2" />
</attr>
</declare-styleable>

</resources>

```

编程使用：

```

CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//然后使用
setStroke(float 宽度, int 颜色, Paint.Join 连接方式, float 斜接长度);
//设置前的方法
setText("示例文本");

```

第4.9节：单个TextView显示两种不同颜色

可以通过传入文本和字体颜色名称给以下函数来创建彩色文本：

```

private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}

```

然后，可以使用下面的示例代码将彩色文本设置到TextView（甚至Button、EditText等）中。

首先，定义一个TextView，如下所示：

```
TextView txtView = (TextView) findViewById(R.id.txtView);
```

然后，创建不同颜色的文本并赋值给字符串：

```

String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel", "#000080");

```

最后，将两个不同颜色的字符串设置到TextView中：

```
txtView.setText(Html.fromHtml(name + surName));
```

参考截图：

```

<attr name="outerShadowRadius" format="dimension" />
<attr name="strokeWidth" format="dimension" />
<attr name="strokeMiter" format="dimension" />
<attr name="strokeColor" format="color" />
<attr name="strokeJoinStyle">
    <enum name="miter" value="0" />
    <enum name="bevel" value="1" />
    <enum name="round" value="2" />
</attr>
</declare-styleable>

</resources>

```

Programmatically usage:

```

CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//then use
setStroke(float width, int color, Paint.Join join, float miter);
//method before setting
setText("Sample Text");

```

Section 4.9: Single TextView with two different colors

Colored text can be created by passing the text and a font color name to the following function:

```

private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}

```

The colored text can then be set to a TextView (or even to a Button, EditText, etc.) by using the example code below.

First, define a TextView as follows:

```
TextView txtView = (TextView) findViewById(R.id.txtView);
```

Then, create differently colored text and assign it to strings:

```

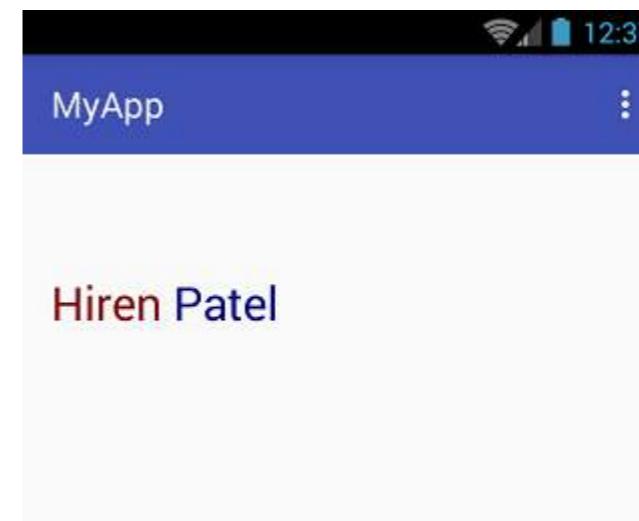
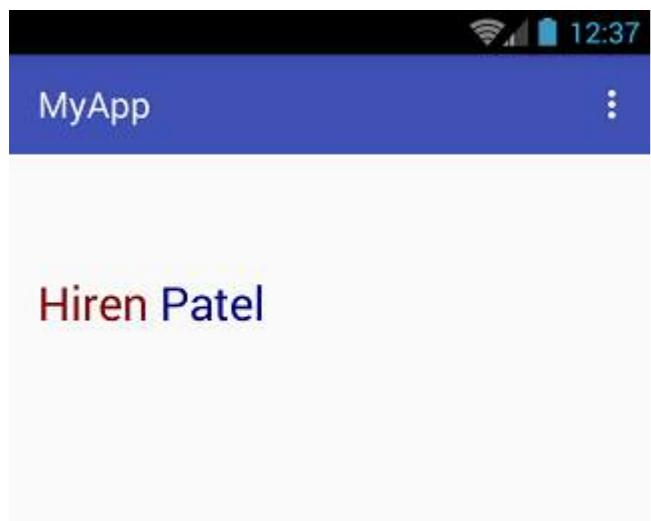
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel", "#000080");

```

Finally, set the two differently colored strings to the TextView:

```
txtView.setText(Html.fromHtml(name + surName));
```

Reference screenshot:



第5章：自动完成文本视图（AutoCompleteTextView）

第5.1节：使用CustomAdapter、ClickListener和Filter的自动完成

主布局 : activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <AutoCompleteTextView
        android:id="@+id/auto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="2"
        android:hint="@string/hint_enter_name" />
</LinearLayout>
```

行布局 row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="中等文本"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">请输入姓名</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Chapter 5: AutoCompleteTextView

Section 5.1: AutoComplete with CustomAdapter, ClickListener and Filter

Main layout : activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <AutoCompleteTextView
        android:id="@+id/auto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="2"
        android:hint="@string/hint_enter_name" />
</LinearLayout>
```

Row layout row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">Enter Name</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```

setContentView(R.layout.activity_main);
    mList = retrievePeople();
txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
    adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
    txtSearch.setAdapter(adapter);
txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id) {
            //this is the way to find selected object/item
selectedPerson = (People) adapterView.getItemAtPosition(pos);
        }
    });

private List<People> retrievePeople() {
    List<People> list = new ArrayList<People>();
    list.add(new People("詹姆斯", "邦德", 1));
    list.add(new People("杰森", "伯恩", 2));
    list.add(new People("伊桑", "亨特", 3));
    list.add(new People("福尔摩斯", "夏洛克", 4));
    list.add(new People("大卫", "贝克汉姆", 5));
    list.add(new People("布莱恩", "亚当斯", 6));
    list.add(new People("阿尔扬", "罗本", 7));
    list.add(new People("范", "佩西", 8));
    list.add(new People("齐内丁", "齐达内", 9));
    list.add(new People("路易斯", "菲戈", 10));
    list.add(new People("约翰", "沃森", 11));
    return list;
}
}

```

Model class : People.java

```

public class People {

    private String 名字, 姓氏;
    private int 身份证号;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLastname() {

```

```

setContentView(R.layout.activity_main);
    mList = retrievePeople();
txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
    adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
    txtSearch.setAdapter(adapter);
txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id) {
            //this is the way to find selected object/item
selectedPerson = (People) adapterView.getItemAtPosition(pos);
        }
    });

private List<People> retrievePeople() {
    List<People> list = new ArrayList<People>();
    list.add(new People("James", "Bond", 1));
    list.add(new People("Jason", "Bourne", 2));
    list.add(new People("Ethan", "Hunt", 3));
    list.add(new People("Sherlock", "Holmes", 4));
    list.add(new People("David", "Beckham", 5));
    list.add(new People("Bryan", "Adams", 6));
    list.add(new People("Arjen", "Robben", 7));
    list.add(new People("Van", "Persie", 8));
    list.add(new People("Zinedine", "Zidane", 9));
    list.add(new People("Luis", "Figo", 10));
    list.add(new People("John", "Watson", 11));
    return list;
}
}

```

Model class : People.java

```

public class People {

    private String name, lastName;
    private int id;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLastname() {

```

```

        return lastName;
    }

    public void setlastName(String lastName) {
        this.lastName = lastName;
    }
}

```

适配器类 : PeopleAdapter.java

```

public class PeopleAdapter extends ArrayAdapter<People> {

    Context context;
    int resource, textViewResourceId;
    List<People> items, tempItems, suggestions;

    public PeopleAdapter(Context context, int resource, int textViewResourceId, List<People> items)
    {
        super(context, resource, textViewResourceId, items);
        this.context = context;
        this.resource = resource;
        this.textViewResourceId = textViewResourceId;
        this.items = items;
        tempItems = new ArrayList<People>(items); // 这就是区别所在。
        suggestions = new ArrayList<People>();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = convertView;
        if (convertView == null) {
            LayoutInflator inflater = (LayoutInflator)
            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inflater.inflate(R.layout.row, parent, false);
        }
        People people = items.get(position);
        if (people != null) {
            TextView lblName = (TextView) view.findViewById(R.id.lbl_name);
            if (lblName != null)
                lblName.setText(people.getName());
        }
        return view;
    }

    @Override
    public Filter getFilter() {
        return nameFilter;
    }

    /**
     * 自定义过滤器实现，用于我们提供的自定义建议。
     */
    过滤器 nameFilter = new 过滤器() {
        @Override
        public CharSequence convertResultToString(Object resultValue) {
            String str = ((People) resultValue).getName();
            return str;
        }

        @Override
        protected FilterResults performFiltering(CharSequence constraint) {

```

```

        return lastName;
    }

    public void setlastName(String lastName) {
        this.lastName = lastName;
    }
}

```

Adapter class : PeopleAdapter.java

```

public class PeopleAdapter extends ArrayAdapter<People> {

    Context context;
    int resource, textViewResourceId;
    List<People> items, tempItems, suggestions;

    public PeopleAdapter(Context context, int resource, int textViewResourceId, List<People> items)
    {
        super(context, resource, textViewResourceId, items);
        this.context = context;
        this.resource = resource;
        this.textViewResourceId = textViewResourceId;
        this.items = items;
        tempItems = new ArrayList<People>(items); // this makes the difference.
        suggestions = new ArrayList<People>();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = convertView;
        if (convertView == null) {
            LayoutInflator inflater = (LayoutInflator)
            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inflater.inflate(R.layout.row, parent, false);
        }
        People people = items.get(position);
        if (people != null) {
            TextView lblName = (TextView) view.findViewById(R.id.lbl_name);
            if (lblName != null)
                lblName.setText(people.getName());
        }
        return view;
    }

    @Override
    public Filter getFilter() {
        return nameFilter;
    }

    /**
     * Custom Filter implementation for custom suggestions we provide.
     */
    Filter nameFilter = new Filter() {
        @Override
        public CharSequence convertResultToString(Object resultValue) {
            String str = ((People) resultValue).getName();
            return str;
        }

        @Override
        protected FilterResults performFiltering(CharSequence constraint) {

```

```

    if (constraint != null) {
        suggestions.clear();
        for (People people : tempItems) {
            if
                (people.getName().toLowerCase().contains(constraint.toString().toLowerCase())))
                    suggestions.add(people);
        }
    }
    FilterResults filterResults = new FilterResults();
    filterResults.values = suggestions;
    filterResults.count = suggestions.size();
    return filterResults;
} else {
    return new FilterResults();
}
}

@Override
protected void publishResults(CharSequence constraint, FilterResults results) {
    List<People> filterList = (ArrayList<People>) results.values;
    if (results != null && results.count > 0) {
        clear();
        for (People people : filterList) {
            add(people);
            notifyDataSetChanged();
        }
    }
}
}
}

```

```

    if (constraint != null) {
        suggestions.clear();
        for (People people : tempItems) {
            if
                (people.getName().toLowerCase().contains(constraint.toString().toLowerCase())))
                    suggestions.add(people);
        }
    }
    FilterResults filterResults = new FilterResults();
    filterResults.values = suggestions;
    filterResults.count = suggestions.size();
    return filterResults;
} else {
    return new FilterResults();
}
}

@Override
protected void publishResults(CharSequence constraint, FilterResults results) {
    List<People> filterList = (ArrayList<People>) results.values;
    if (results != null && results.count > 0) {
        clear();
        for (People people : filterList) {
            add(people);
            notifyDataSetChanged();
        }
    }
}
}

```

第5.2节：简单的硬编码自动完成文本视图

设计（布局XML）：

```

<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="65dp"
    android:ems="10" />

```

在代码中通过setContentView()（或其片段或自定义视图的等效方法）查找视图：

```

final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);

```

通过适配器提供硬编码数据：

```

String[] countries = getResources().getStringArray(R.array.list_of_countries);
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries);
myAutoCompleteTextView.setAdapter(adapter);

```

提示：虽然首选方式是通过某种Loader提供数据，而不是像这样使用硬编码列表。

Section 5.2: Simple, hard-coded AutoCompleteTextView

Design (layout XML):

```

<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="65dp"
    android:ems="10" />

```

Find the view in code after setContentView() (or its fragment or custom view equivalent):

```

final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);

```

Provide hard-coded data via an adapter:

```

String[] countries = getResources().getStringArray(R.array.list_of_countries);
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries);
myAutoCompleteTextView.setAdapter(adapter);

```

Tip: Though the preferred way would be to provide data via a Loader of some kind instead of a hard-coded list like this.

第6章：自动调整大小的TextView

一个可以自动调整文本大小以完美适应其边界的TextView。

Android O 允许你指示 TextView 根据 TextView 的特性和边界，自动调整文本大小以填充其布局，文本大小可以自动扩大或缩小。

您可以在代码或XML中设置TextView的自动缩放。

设置自动缩放TextView有两种方式：粒度和预设尺寸

第6.1节：粒度

在Java中：

调用[setAutoSizeTextTypeUniformWithConfiguration\(\)方法](#)：

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize, int autoSizeStepGranularity, int unit)
```

在XML中：

使用autoSizeMinTextSize、autoSizeMaxTextSize和autoSizeStepGranularity属性在布局XML文件中设置自动缩放尺寸：

```
<TextView android:id="@+id/autosizing_textview_presetsize"
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:layout_marginLeft="0dp"
    android:layout_marginTop="0dp"
    android:autoSizeMaxTextSize="100sp"
    android:autoSizeMinTextSize="12sp"
    android:autoSizeStepGranularity="2sp"
    android:autoSizeText="uniform"
    android:text="你好，世界！"
    android:textSize="100sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

请查看 [GitHub 上的AutosizingTextViews-Demo](#)以获取更多详情。

第6.2节：预设尺寸

在Java中：

调用[setAutoSizeTextTypeUniformWithPresetSizes\(\)方法](#)：

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

在XML中：

在布局XML文件中使用autoSizePresetSizes属性：

```
<TextView android:id="@+id/autosizing_textview_presetsize"
```

Chapter 6: Autosizing TextViews

A TextView that automatically resizes text to fit perfectly within its bounds.

Android O allows you to instruct a TextView to let the size of the text expand or contract automatically to fill its layout based on the TextView's characteristics and boundaries.

You can set up the TextView autosizing in either code or XML.

There are two ways to set autosizing TextView: **Granularity** and **Preset Sizes**

Section 6.1: Granularity

In Java:

Call the [setAutoSizeTextTypeUniformWithConfiguration\(\) method](#):

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize, int autoSizeStepGranularity, int unit)
```

In XML:

Use the autoSizeMinTextSize, autoSizeMaxTextSize, and autoSizeStepGranularity attributes to set the auto-sizing dimensions in the layout XML file:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:layout_marginLeft="0dp"
    android:layout_marginTop="0dp"
    android:autoSizeMaxTextSize="100sp"
    android:autoSizeMinTextSize="12sp"
    android:autoSizeStepGranularity="2sp"
    android:autoSizeText="uniform"
    android:text="Hello World!"
    android:textSize="100sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Check out the [AutosizingTextViews-Demo](#) at GitHub for more details.

Section 6.2: Preset Sizes

In Java:

Call the [setAutoSizeTextTypeUniformWithPresetSizes\(\) method](#):

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

In XML:

Use the autoSizePresetSizes attribute in the layout XML file:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
```

```
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeText="uniform"
        android:autoSizePresetSizes="@array/autosize_text_sizes"
        android:text="你好，世界！"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

要将数组作为资源访问，请在res/values/arrays.xml文件中定义数组：

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

请查看 [GitHub 上的AutosizingTextViews-Demo](#)以获取更多详情。

```
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeText="uniform"
        android:autoSizePresetSizes="@array/autosize_text_sizes"
        android:text="Hello World!"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

To access the array as a resource, define the array in the `res/values/arrays.xml` file:

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

Check out the [AutosizingTextViews-Demo](#) at GitHub for more details.

第7章：ListView

ListView是一个视图组，它将来自数组或数据库等数据源的多个项目分组，并以可滚动列表的形式显示它们。数据通过Adapter类与ListView绑定。

第7.1节：自定义ArrayAdapter

默认情况下，ArrayAdapter类通过调用每个项目的toString()方法为每个数组项目创建一个视图，并将内容放入TextView中。

要为每个项目创建复杂视图（例如，如果你想为每个数组项目使用ImageView），请扩展 ArrayAdapter类并重写 getView()方法，以返回你想要的每个项目的视图类型。

例如：

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflater inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public long getItemId(int position)
    {
        //这是一个示例
        YourClassData data = (YourClassData) getItem(position);
        return data.ID;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent)
    {
        ViewHolder viewHolder;
        if (view == null) {
            view = inflater.inflate(R.layout.custom_row_layout_design, null);
            // 进行一些初始化

            //获取项目布局上的视图并设置值。
            viewHolder = new ViewHolder(view);
            view.setTag(viewHolder);
        }
        else {
            viewHolder = (ViewHolder) view.getTag();
        }

        //Retrieve your object
        YourClassData data = (YourClassData) getItem(position);

        viewHolder.txt.setTypeface(m_Font);
        viewHolder.txt.setText(data.text);
        viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

        return view;
    }
}
```

Chapter 7: ListView

ListView is a viewgroup which groups several items from a data source like array or database and displays them in a scrollable list. Data are bound with listview using an Adapter class.

Section 7.1: Custom ArrayAdapter

By default the ArrayAdapter class creates a view for each array item by calling `toString()` on each item and placing the contents in a TextView.

To create a complex view for each item (for example, if you want an ImageView for each array item), extend the ArrayAdapter class and override the `getView()` method to return the type of View you want for each item.

For example:

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflater inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public long getItemId(int position)
    {
        //It is just an example
        YourClassData data = (YourClassData) getItem(position);
        return data.ID;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent)
    {
        ViewHolder viewHolder;
        if (view == null) {
            view = inflater.inflate(R.layout.custom_row_layout_design, null);
            // Do some initialization

            //Retrieve the view on the item layout and set the value.
            viewHolder = new ViewHolder(view);
            view.setTag(viewHolder);
        }
        else {
            viewHolder = (ViewHolder) view.getTag();
        }

        //Retrieve your object
        YourClassData data = (YourClassData) getItem(position);

        viewHolder.txt.setTypeface(m_Font);
        viewHolder.txt.setText(data.text);
        viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

        return view;
    }
}
```

```

private class ViewHolder
{
    private final TextView txt;
    private final ImageView img;

    private ViewHolder(View view)
    {
        txt = (TextView) view.findViewById(R.id.txt);
        img = (ImageView) view.findViewById(R.id.img);
    }
}

```

第7.2节：使用ArrayAdapter的基本ListView

默认情况下，`ArrayAdapter` 会通过调用每个数组项的 `toString()` 方法来为每个数组项创建视图，并将内容放入一个 `TextView` 中。

示例：

```

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, myStringArray);

```

其中 `android.R.layout.simple_list_item_1` 是包含数组中每个字符串的 `TextView` 的布局。

然后只需在你的 `ListView` 上调用 `setAdapter()`：

```

ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);

```

如果想使用除 `TextView` 以外的视图来显示数组内容，例如 `ImageView`，或者想让视图显示除 `toString()` 结果之外的其他数据，可以重写 `getView(int, View, ViewGroup)` 方法，返回你想要的视图类型。请查看此示例。

第7.3节：使用CursorAdapter进行过滤

```

// 获取你的ListView的引用
ListView listResults = (ListView) findViewById(R.id.listResults);

// 设置其适配器
listResults.setAdapter(adapter);

// 启用 ListView 中的过滤功能
listResults.setTextFilterEnabled(true);

// 为过滤准备适配器
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {

        // 在实际应用中，做比字符串拼接更安全的处理
        // 但这取决于你的数据库结构
        // 这是过滤时执行的查询语句
        String query = "SELECT _ID as _id, name FROM MYTABLE "
            + "where name like '%" + constraint + "%' "
            + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
    }
})

```

```

private class ViewHolder
{
    private final TextView txt;
    private final ImageView img;

    private ViewHolder(View view)
    {
        txt = (TextView) view.findViewById(R.id.txt);
        img = (ImageView) view.findViewById(R.id.img);
    }
}

```

Section 7.2: A basic ListView with an ArrayAdapter

By default the `ArrayAdapter` creates a view for each array item by calling `toString()` on each item and placing the contents in a `TextView`.

Example:

```

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, myStringArray);

```

where `android.R.layout.simple_list_item_1` is the layout that contains a `TextView` for each string in the array.

Then simply call `setAdapter()` on your `ListView`:

```

ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);

```

To use something other than `TextViews` for the array display, for instance, `ImageViews`, or to have some of data besides `toString()` results fill the views, override `getView(int, View, ViewGroup)` to return the type of view you want. Check this example.

Section 7.3: Filtering with CursorAdapter

```

// Get the reference to your ListView
ListView listResults = (ListView) findViewById(R.id.listResults);

// Set its adapter
listResults.setAdapter(adapter);

// Enable filtering in ListView
listResults.setTextFilterEnabled(true);

// Prepare your adapter for filtering
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {

        // in real life, do something more secure than concatenation
        // but it will depend on your schema
        // This is the query that will run on filtering
        String query = "SELECT _ID as _id, name FROM MYTABLE "
            + "where name like '%" + constraint + "%' "
            + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
    }
})

```

});

假设你的查询将在用户每次输入EditText时运行：

```
EditText queryText = (EditText) findViewById(R.id.textQuery);
queryText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, final int start, final int count, final
int after) {

}

@Override
public void onTextChanged(CharSequence s, final int start, final int before, final
int count) {
    // 这是过滤器的实际作用
adapter.getFilter().filter(s.toString());
    // 别忘了通知适配器
adapter.notifyDataSetChanged();
}

@Override
public void afterTextChanged(Editable s) {
}

});
```

});

Let's say your query will run every time the user types in an EditText:

```
EditText queryText = (EditText) findViewById(R.id.textQuery);
queryText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, final int start, final int count, final
int after) {

}

@Override
public void onTextChanged(CharSequence s, final int start, final int before, final
int count) {
    // This is the filter in action
adapter.getFilter().filter(s.toString());
    // Don't forget to notify the adapter
adapter.notifyDataSetChanged();
}

@Override
public void afterTextChanged(Editable s) {
}

});
```

第8章：布局

布局定义了用户界面的视觉结构，例如一个活动或小部件。

布局在XML中声明，包括将出现在其中的屏幕元素。可以向应用程序添加代码，以在运行时修改屏幕对象的状态，包括那些在XML中声明的对象。

第8.1节：LayoutParams

每个ViewGroup（例如LinearLayout、RelativeLayout、CoordinatorLayout等）都需要存储关于其子元素属性的信息，即其子元素在ViewGroup中的布局方式。这些信息存储在一个包装类ViewGroup.LayoutParams的对象中。

为了包含特定布局类型的参数，ViewGroups使用ViewGroup.LayoutParams类的子类。

例如，对于

- LinearLayout是LinearLayout.LayoutParams
- RelativeLayout是RelativeLayout.LayoutParams
- CoordinatorLayout是CoordinatorLayout.LayoutParams
- ...

大多数ViewGroups重用为其子元素设置margins的功能，因此它们不直接继承ViewGroup.LayoutParams，而是继承ViewGroup.MarginLayoutParams（它本身是ViewGroup.LayoutParams的子类）。

XML中的LayoutParams

LayoutParams对象是基于膨胀的布局xml文件创建的。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_gravity="right"
        android:gravity="bottom"
        android:text="示例文本"
        android:textColor="@android:color/holo_green_dark"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_green_dark"
        android:scaleType="centerInside"
        android:src="@drawable/example"/>

</LinearLayout>
```

所有以layout_开头的参数指定了**enclosing**布局的工作方式。当布局被填充时，这些参数会被封装在一个合适的LayoutParams对象中，随后该对象将被Layout用来正确地

Chapter 8: Layouts

A layout defines the visual structure for a user interface, such as an activity or widget.

A layout is declared in XML, including screen elements that will appear in it. Code can be added to the application to modify the state of screen objects at runtime, including those declared in XML.

Section 8.1: LayoutParams

Every single ViewGroup (e.g. LinearLayout, RelativeLayout, CoordinatorLayout, etc.) needs to store information about its children's properties. About the way its children are being laid out in the ViewGroup. This information is stored in objects of a wrapper class ViewGroup.LayoutParams.

To include parameters specific to a particular layout type, ViewGroups use subclasses of ViewGroup.LayoutParams class.

E.g. for

- LinearLayout it's LinearLayout.LayoutParams
- RelativeLayout it's RelativeLayout.LayoutParams
- CoordinatorLayout it's CoordinatorLayout.LayoutParams
- ...

Most of ViewGroups reutilize the ability to set margins for their children, so they do not subclass ViewGroup.LayoutParams directly, but they subclass ViewGroup.MarginLayoutParams instead (which itself is a subclass of ViewGroup.LayoutParams).

LayoutParams in xml

LayoutParams objects are created based on the inflated layout xml file.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_gravity="right"
        android:gravity="bottom"
        android:text="Example text"
        android:textColor="@android:color/holo_green_dark"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_green_dark"
        android:scaleType="centerInside"
        android:src="@drawable/example"/>

</LinearLayout>
```

All parameters that begin with layout_ specify how the **enclosing** layout should work. When the layout is inflated, those parameters are wrapped in a proper LayoutParams object, that later will be used by the Layout to properly

在ViewGroup中定位特定的View。一个View的其他属性是直接与View相关的，由View自身处理。

对于TextView：

- layout_width、layout_height和layout_gravity将存储在LinearLayout.LayoutParams对象中，并由LinearLayout使用
- gravity、text和textColor将由TextView自身使用

对于ImageView：

- layout_width、layout_height和layout_weight将存储在LinearLayout.LayoutParams对象中，并由LinearLayout使用
- background、scaleType和src将由ImageView自身使用

获取LayoutParams对象

`getLayoutParams`是View的一个方法，用于获取当前的LayoutParams对象。

因为LayoutParams对象与enclosingViewGroup直接相关，只有当View附加到ViewGroup时，此方法才会返回非空值。你需要记住，这个对象可能并不总是存在。特别是你不应该依赖它存在于View的构造函数中。

```
public class ExampleView extends View {  
  
    public ExampleView(Context context) {  
        super(context);  
        setupView(context);  
    }  
  
    public ExampleView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
        setupView(context);  
    }  
  
    public ExampleView(Context context, AttributeSet attrs, int defStyle) {  
        super(context, attrs, defStyle);  
        setupView(context);  
    }  
  
    private void setupView(Context context) {  
        if (getLayoutParams().height == 50){ // 不要这样做！  
            // 这可能会导致空指针异常  
        doSomething();  
    }  
    }  
    //...  
}
```

如果您想依赖于拥有LayoutParams对象，应该改用onAttachedToWindow方法。

```
public class ExampleView extends View {  
  
    public ExampleView(Context context) {  
        super(context);  
    }
```

position a particular View within the ViewGroup. Other attributes of a View are directly View-related and are processed by the View itself.

For TextView:

- layout_width, layout_height and layout_gravity will be stored in a LinearLayout.LayoutParams object and used by the LinearLayout
- gravity, text and textColor will be used by the TextView itself

For ImageView:

- layout_width, layout_height and layout_weight will be stored in a LinearLayout.LayoutParams object and used by the LinearLayout
- background, scaleType and src will be used by the ImageView itself

Getting LayoutParams object

`getLayoutParams` is a View's method that allows to retrieve a current LayoutParams object.

Because the LayoutParams object is directly related to the **enclosing** ViewGroup, this method will return a non-null value only when View is attached to the ViewGroup. You need to bare in mind that this object might not be present at all times. Especially you should not depend on having it inside View's constructor.

```
public class ExampleView extends View {  
  
    public ExampleView(Context context) {  
        super(context);  
        setupView(context);  
    }  
  
    public ExampleView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
        setupView(context);  
    }  
  
    public ExampleView(Context context, AttributeSet attrs, int defStyle) {  
        super(context, attrs, defStyle);  
        setupView(context);  
    }  
  
    private void setupView(Context context) {  
        if (getLayoutParams().height == 50){ // DO NOT DO THIS!  
            // This might produce NullPointerException  
        doSomething();  
    }  
    }  
    //...  
}
```

If you want to depend on having LayoutParams object, you should use `onAttachedToWindow` method instead.

```
public class ExampleView extends View {  
  
    public ExampleView(Context context) {  
        super(context);  
    }
```

```

public ExampleView(Context context, AttributeSet attrs) {
    super(context, attrs);
}

public ExampleView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}

@Override
protected void onAttachedToWindow() {
    super.onAttachedToWindow();
    if (getLayoutParams().height == 50) { // getLayoutParams() 在这里不会返回 null
        doSomething();
    }
}

//...
}

```

类型转换 LayoutParams 对象

你可能需要使用特定于某个 ViewGroup 的功能（例如，你可能想要以编程方式更改 RelativeLayout 的规则）。为此，你需要知道如何正确地转换 ViewGroup.LayoutParams 对象。

当获取一个子 View 的 LayoutParams 对象，而该子视图实际上是另一个

ViewGroup 时，这可能会有点令人困惑。

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/outer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/inner_layout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="right"/>

</LinearLayout>

```

重要提示：LayoutParams对象的类型与ENCLOSINGViewGroup的类型直接相关。

错误的类型转换：

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
// 错误！这将导致 ClassCastException

```

正确的类型转换：

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
// 正确！外层布局是 LinearLayout

```

```

public ExampleView(Context context, AttributeSet attrs) {
    super(context, attrs);
}

public ExampleView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}

@Override
protected void onAttachedToWindow() {
    super.onAttachedToWindow();
    if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
        doSomething();
    }
}

//...
}

```

Casting LayoutParams object

You might need to use features that are specific to a particular ViewGroup (e.g. you might want to programmatically change rules of a RelativeLayout). For that purpose you will need to know how to properly cast the ViewGroup.LayoutParams object.

This might be a bit confusing when getting a LayoutParams object for a child View that actually is another ViewGroup.

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/outer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/inner_layout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="right"/>

</LinearLayout>

```

IMPORTANT: The type of LayoutParams object is directly related to the type of the **ENCLOSING** ViewGroup.

Incorrect casting:

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
// INCORRECT! This will produce ClassCastException

```

Correct casting:

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
// CORRECT! the enclosing layout is a LinearLayout

```

第8.2节：Gravity 和 layout gravity

android:layout_gravity

- android:layout_gravity 用于设置元素在其父容器中的位置（例如，布局中的子View）。
- 由 [LinearLayout](#) 和 [FrameLayout](#) 支持

android:gravity

- android:gravity 用于设置元素内部内容的位置（例如，TextView中的文本）。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="left"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimary"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimary"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimary"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="center"
        android:gravity="center_vertical">

    </LinearLayout>

```

Section 8.2: Gravity and layout gravity

android:layout_gravity

- android:layout_gravity is used to set the position of an element in its parent (e.g. a child View inside a Layout).
- Supported by [LinearLayout](#) and [FrameLayout](#)

android:gravity

- android:gravity is used to set the position of content inside an element (e.g. a text inside a TextView).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="left"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimary"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimary"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimary"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="center"
        android:gravity="center_vertical">

    </LinearLayout>

```

```

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/first"
    android:background="@color/colorAccent"
    android:gravity="left"/>

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/second"
    android:background="@color/colorAccent"
    android:gravity="center"/>

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/third"
    android:background="@color/colorAccent"
    android:gravity="right"/>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:layout_gravity="right"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimaryDark"
        android:gravity="left"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimaryDark"
        android:gravity="center"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimaryDark"
        android:gravity="right"/>

</LinearLayout>

</LinearLayout>

```

渲染结果如下：

```

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/first"
    android:background="@color/colorAccent"
    android:gravity="left"/>

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/second"
    android:background="@color/colorAccent"
    android:gravity="center"/>

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/third"
    android:background="@color/colorAccent"
    android:gravity="right"/>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:layout_gravity="right"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimaryDark"
        android:gravity="left"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimaryDark"
        android:gravity="center"/>

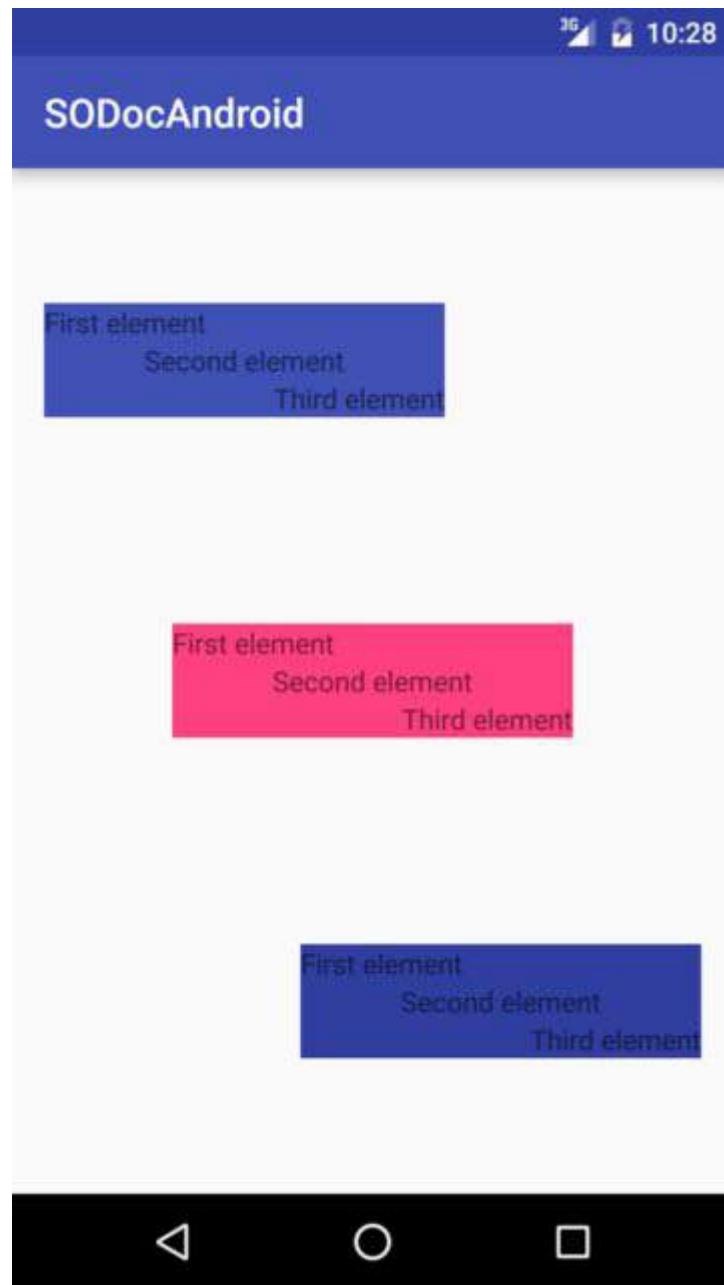
    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimaryDark"
        android:gravity="right"/>

</LinearLayout>

</LinearLayout>

```

Which gets rendered as following:



第8.3节：CoordinatorLayout滚动行为

版本 ≥ 2.3-2.3.2

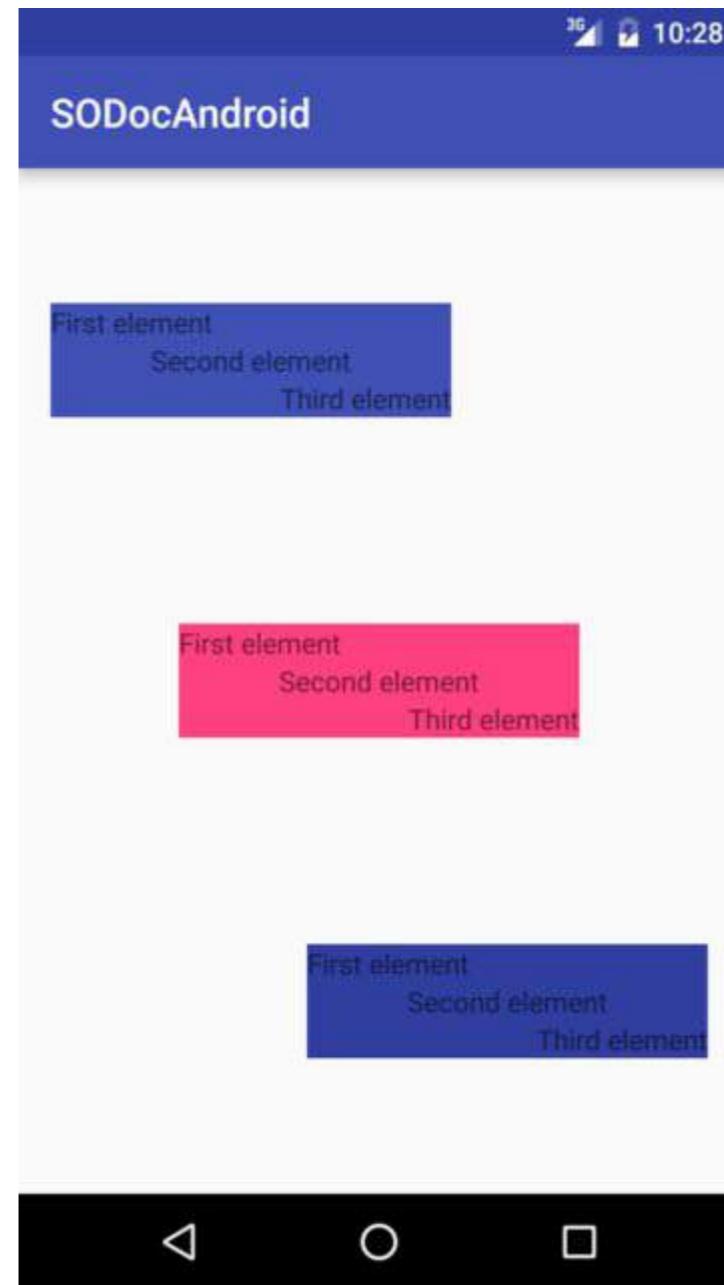
可以使用一个外层的CoordinatorLayout来实现Material Design滚动效果，前提是内部布局支持嵌套滚动（Nested Scrolling），例如NestedScrollView或RecyclerView。

例如：

- app:layout_scrollFlags="scroll|enterAlways" 用于工具栏属性中
- app:layout_behavior="@string/appbar_scrolling_view_behavior" 用于ViewPager属性中
- RecyclerView用于ViewPager的Fragment中

这是Activity中使用的布局xml文件：

```
<android.support.design.widget.CoordinatorLayout  
    android:id="@+id/main_layout"  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"
```



Section 8.3: CoordinatorLayout Scrolling Behavior

Version ≥ 2.3-2.3.2

An enclosing CoordinatorLayout can be used to achieve [Material Design Scrolling Effects](#) when using inner layouts that support Nested Scrolling, such as [NestedScrollView](#) or [RecyclerView](#).

For this example:

- app:layout_scrollFlags="scroll|enterAlways" is used in the Toolbar properties
- app:layout_behavior="@string/appbar_scrolling_view_behavior" is used in the ViewPager properties
- A RecyclerView is used in the ViewPager Fragments

Here is the layout xml file used in an Activity:

```
<android.support.design.widget.CoordinatorLayout  
    android:id="@+id/main_layout"  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"
```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:elevation="0dp"
            app:layout_scrollFlags="scroll|enterAlways"
        />

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
    />

</android.support.design.widget.AppBarLayout>

<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_below="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
/>

</android.support.design.widget.CoordinatorLayout>

```

结果：

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:elevation="0dp"
            app:layout_scrollFlags="scroll|enterAlways"
        />

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
    />

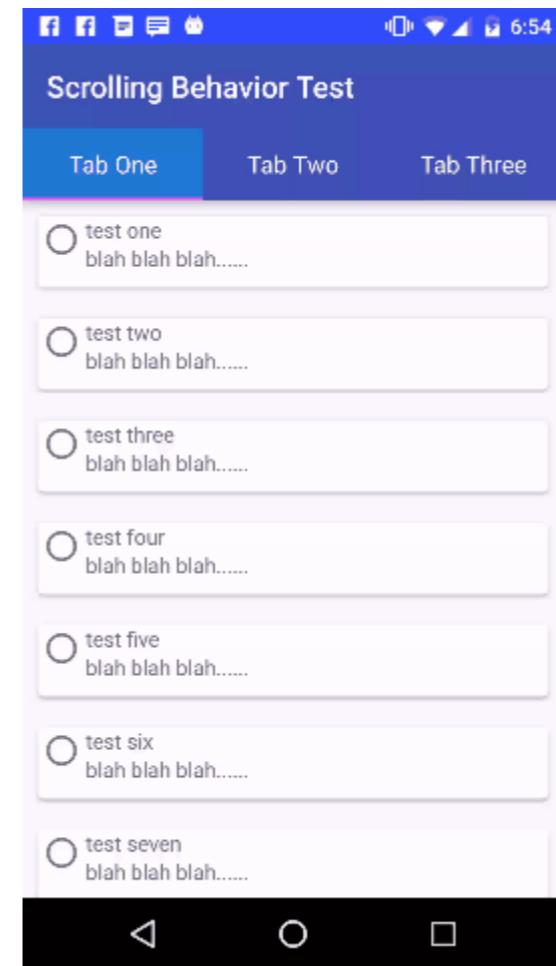
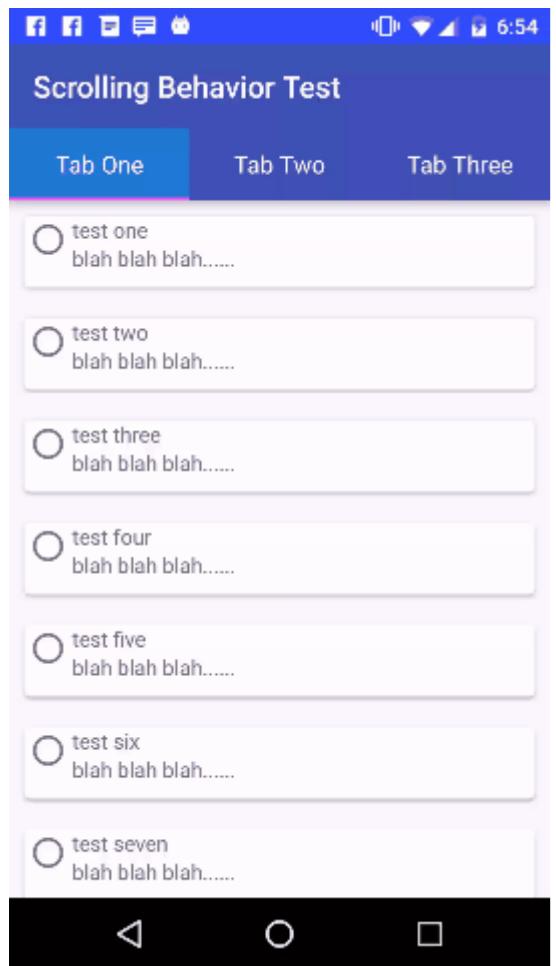
</android.support.design.widget.AppBarLayout>

<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_below="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
/>

</android.support.design.widget.CoordinatorLayout>

```

Result:



第8.4节：百分比布局

版本 ≥ 2.3

Percent Support Library 提供了 PercentFrameLayout 和 PercentRelativeLayout 两个 ViewGroup，它们提供了一种简单的方法来以整体尺寸的百分比来指定 View 的尺寸和边距。

您可以通过在依赖项中添加以下内容来使用 Percent Support Library。

```
compile 'com.android.support:percent:25.3.1'
```

如果您想显示一个水平方向填满屏幕但垂直方向只占屏幕一半的视图，可以按如下方式操作。

```
<android.support.percent.PercentFrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
<FrameLayout  
    app:layout_widthPercent="100%"  
    app:layout_heightPercent="50%"  
    android:background="@android:color/black" />  
  
<android.support.percent.PercentFrameLayout>
```

你也可以在单独的XML文件中定义百分比，例如：

Section 8.4: Percent Layouts

Version ≥ 2.3

The Percent Support Library provides PercentFrameLayout and PercentRelativeLayout, two ViewGroups that provide an easy way to specify View **dimensions and margins** in terms of a **percentage** of the overall size.

You can use the Percent Support Library by adding the following to your dependencies.

```
compile 'com.android.support:percent:25.3.1'
```

If you wanted to display a view that fills the screen horizontally but only half the screen vertically you would do the following.

```
<android.support.percent.PercentFrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
<FrameLayout  
    app:layout_widthPercent="100%"  
    app:layout_heightPercent="50%"  
    android:background="@android:color/black" />  
  
<android.support.percent.PercentFrameLayout>
```

You can also define the percentages in a separate XML file with code such as:

```
<fraction name="margin_start_percent">25%</fraction>
```

并在布局中通过@fraction/margin_start_percent引用它们。

它们还包含通过app:layout_aspectRatio设置自定义宽高比的功能。

这允许你只设置单一维度，比如只设置宽度，高度将根据你定义的宽高比自动确定，无论是4:3、16:9，还是正方形的1:1宽高比。

例如：

```
<ImageView  
    app:layout_widthPercent="100%"  
    app:layout_aspectRatio="178%"  
    android:scaleType="centerCrop"  
    android:src="@drawable/header_background" />
```

第8.5节：视图权重

LinearLayout中最常用的属性之一是其子视图的weight。权重定义了一个视图相对于LinearLayout中其他视图将占用多少空间。

当你想给某个组件相对于其他组件分配特定屏幕空间时，会使用权重。

关键属性：

- weightSum 是所有子视图权重的总和。如果你没有指定weightSum，系统将自动计算所有权重的总和。
- layout_weight 指定控件将占用的总权重中的空间比例。

代码：

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/activity_main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal"  
    android:weightSum="4">  
  
<EditText  
    android:layout_weight="2"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:text="在此输入文本" />  
  
<Button  
    android:layout_weight="1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:text="Text1" />  
  
<Button  
    android:layout_weight="1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"
```

```
<fraction name="margin_start_percent">25%</fraction>
```

And refer to them in your layouts with @fraction/margin_start_percent.

They also contain the ability to set a custom **aspect ratio** via app:layout_aspectRatio.

This allows you to set only a single dimension, such as only the width, and the height will be automatically determined based on the aspect ratio you've defined, whether it is 4:3 or 16:9 or even a square 1:1 aspect ratio.

For example:

```
<ImageView  
    app:layout_widthPercent="100%"  
    app:layout_aspectRatio="178%"  
    android:scaleType="centerCrop"  
    android:src="@drawable/header_background" />
```

Section 8.5: View Weight

One of the most used attribute for LinearLayout is the weight of its child views. Weight defines how much space a view will consume compared to other views within a LinearLayout.

Weight is used when you want to give specific screen space to one component compared to other.

Key Properties:

- weightSum is the overall sum of weights of all child views. If you don't specify the weightSum, the system will calculate the sum of all the weights on its own.
- layout_weight specifies the amount of space out of the total weight sum the widget will occupy.

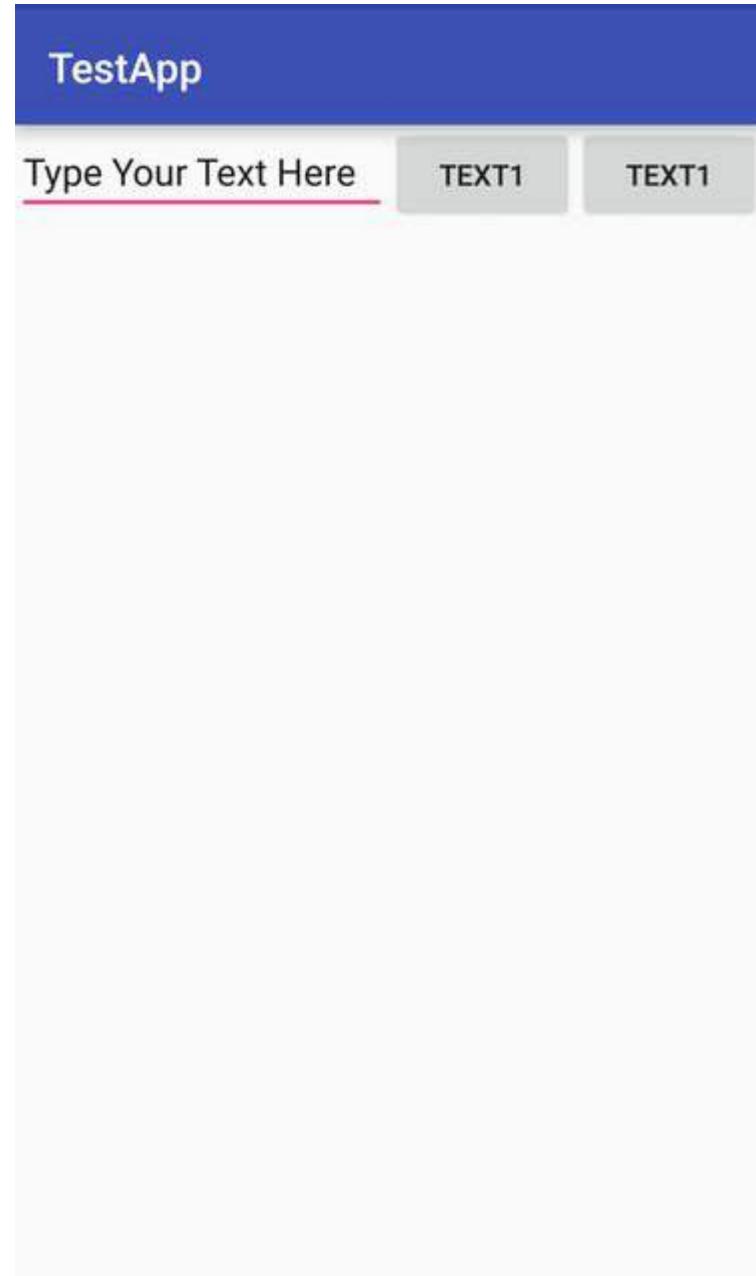
Code:

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/activity_main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal"  
    android:weightSum="4">  
  
<EditText  
    android:layout_weight="2"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:text="Type Your Text Here" />  
  
<Button  
    android:layout_weight="1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:text="Text1" />  
  
<Button  
    android:layout_weight="1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"
```

```
    android:text="Text1" />
```

```
</LinearLayout>
```

输出为：



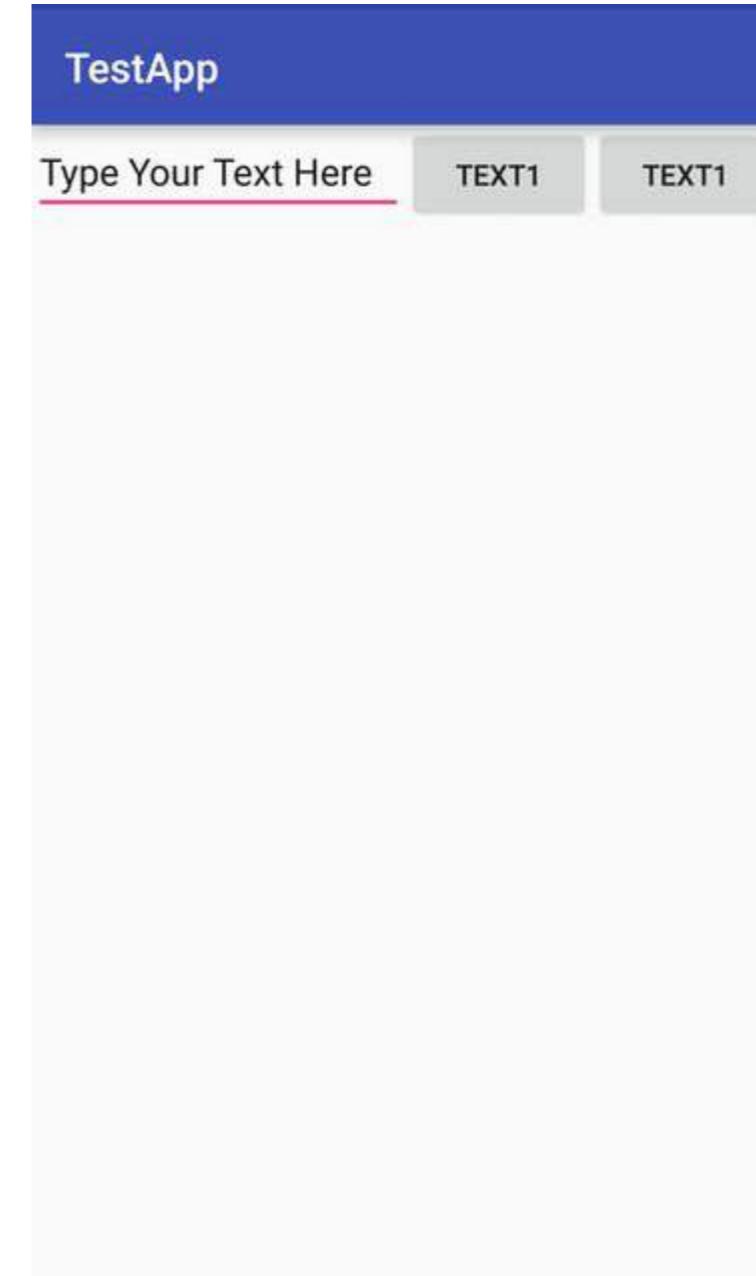
现在即使设备尺寸更大，EditText 也会占据屏幕空间的 2/4。因此，您的应用在所有屏幕上的外观保持一致。

注意：这里的 layout_width 设置为 0dp，因为控件空间是水平分割的。如果控件需要垂直排列，layout_height 将设置为 0dp。这样做是为了提高代码效率，因为在运行时系统不会尝试分别计算宽度或高度，这由权重管理。如果您改用 wrap_content，系统会先尝试计算宽度/高度，然后再应用权重属性，这会导致额外的计算过程。

```
    android:text="Text1" />
```

```
</LinearLayout>
```

The output is:



Now even if the size of the device is larger, the EditText will take 2/4 of the screen's space. Hence the look of your app is seen consistent across all screens.

Note: Here the layout_width is kept 0dp as the widget space is divided horizontally. If the widgets are to be aligned vertically layout_height will be set to 0dp. This is done to increase the efficiency of the code because at runtime the system won't attempt to calculate the width or height respectively as this is managed by the weight. If you instead used wrap_content the system would attempt to calculate the width/height first before applying the weight attribute which causes another calculation cycle.

第8.6节：编程方式创建LinearLayout

层级结构

Section 8.6: Creating LinearLayout programmatically

Hierarchy

- LinearLayout(水平)
 - ImageView
- 线性布局(垂直)
 - 文本视图
 - 文本视图

代码

```
LinearLayout rootView = new LinearLayout(context);
rootView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
rootView.setOrientation(LinearLayout.水平);

// 用于图像视图
ImageView imageView = new ImageView(context);
// 用于水平线性布局
LinearLayout linearLayout2 = new LinearLayout(context);
linearLayout2.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
linearLayout2.setOrientation(LinearLayout.垂直);

TextView tv1 = new TextView(context);
TextView tv2 = new TextView(context);
// 向水平线性布局中添加两个文本视图
linearLayout2.addView(tv1);
linearLayout2.addView(tv2);

// 最后，将图像视图和水平线性布局添加到垂直线性布局（根视图）中
rootView.addView(imageView);
rootView.addView(linearLayout2);
```

第8.7节：LinearLayout（线性布局）

LinearLayout是一个ViewGroup，用于将其子视图排列成单列或单行。方向可以通过调用方法setOrientation()或使用xml属性android:orientation来设置。

1. 垂直方向: android:orientation="vertical"

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/cancel" />

</LinearLayout>
```

以下是该效果的截图：

- LinearLayout(horizontal)
 - ImageView
 - LinearLayout(vertical)
 - TextView
 - TextView

Code

```
LinearLayout rootView = new LinearLayout(context);
rootView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
rootView.setOrientation(LinearLayout.HORIZONTAL);

// for imageview
ImageView imageView = new ImageView(context);
// for horizontal linearlayout
LinearLayout linearLayout2 = new LinearLayout(context);
linearLayout2.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
linearLayout2.setOrientation(LinearLayout.VERTICAL);

TextView tv1 = new TextView(context);
TextView tv2 = new TextView(context);
// add 2 textView to horizontal linearlayout
linearLayout2.addView(tv1);
linearLayout2.addView(tv2);

// finally, add imageview and horizontal linearlayout to vertical linearlayout (rootView)
rootView.addView(imageView);
rootView.addView(linearLayout2);
```

Section 8.7: LinearLayout

The [LinearLayout](#) is a ViewGroup that arranges its children in a single column or a single row. The orientation can be set by calling the method [setOrientation\(\)](#) or using the xml attribute [android:orientation](#).

1. Vertical orientation : android:orientation="vertical"

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/cancel" />

</LinearLayout>
```

Here is a screenshot how this will look like:



2. 水平方向 : android:orientation="horizontal"

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/app_name" />  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@android:string/cancel" />
```

LinearLayout 还支持通过 android:layout_weight

属性为各个子视图分配权重。

第8.8节 : RelativeLayout

RelativeLayout 是一个 ViewGroup，用于以相对位置显示子视图。默认情况下，所有子视图都绘制在



2. Horizontal orientation : android:orientation="horizontal"

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/app_name" />  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@android:string/cancel" />
```

The LinearLayout also supports assigning a weight to individual children with the android:layout_weight attribute.

Section 8.8: RelativeLayout

[RelativeLayout](#) is a ViewGroup that displays child views in relative positions. By default, all child views are drawn at

布局的左上角，因此必须使用 `RelativeLayout.LayoutParams` 提供的各种布局属性来定义每个视图的位置。每个布局属性的值要么是一个布尔值，用于启用相对于父 `RelativeLayout` 的布局位置，要么是一个 ID，引用布局中另一个视图，视图应相对于该视图定位。

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@mipmap/ic_launcher" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_toRightOf="@+id/imageView"
        android:layout_toEndOf="@+id/imageView"
        android:hint="@string/hint" />

</RelativeLayout>
```

以下是该效果的截图：

the top-left of the layout, so you must define the position of each view using the various layout properties available from `RelativeLayout.LayoutParams`. The value for each layout property is either a boolean to enable a layout position relative to the parent `RelativeLayout` or an ID that references another view in the layout against which the view should be positioned.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@mipmap/ic_launcher" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_toRightOf="@+id/imageView"
        android:layout_toEndOf="@+id/imageView"
        android:hint="@string/hint" />

</RelativeLayout>
```

Here is a screenshot how this will look like:



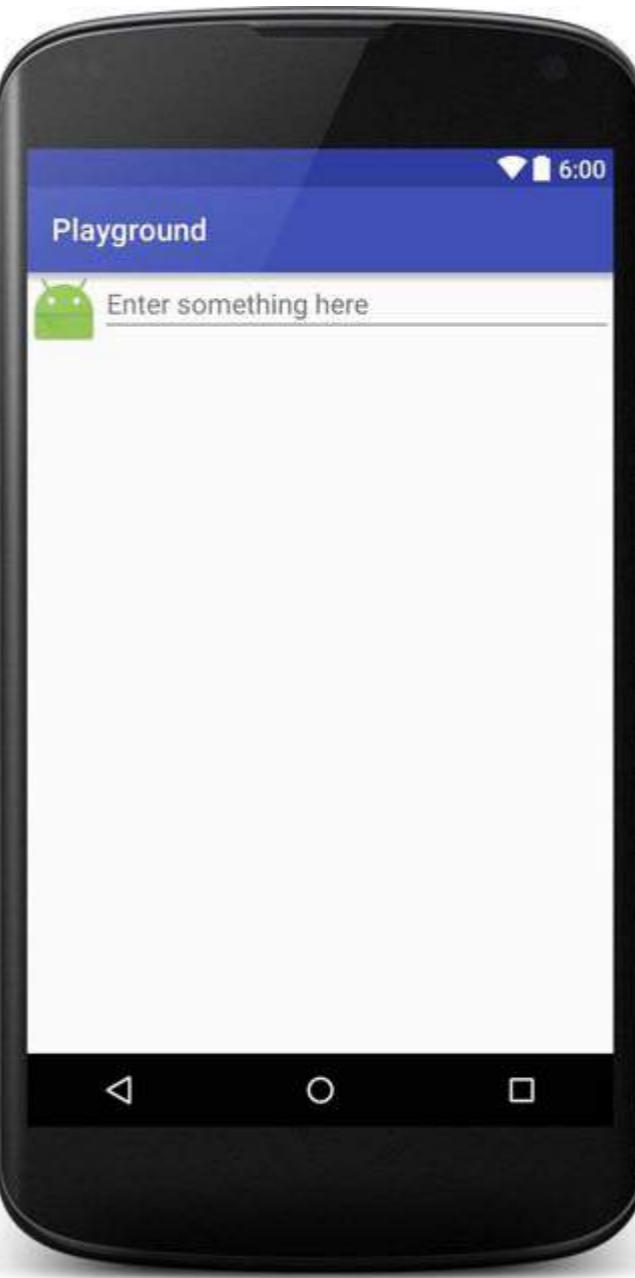
第8.9节：FrameLayout

FrameLayout 旨在屏幕上划定一个区域以显示单个项目。然而，你可以向 FrameLayout 添加多个子视图，并通过为每个子视图分配重力属性，使用 `android:layout_gravity` 属性来控制它们在 FrameLayout 中的位置。

通常，FrameLayout 用于容纳单个子视图。常见用例包括在 Activity 中填充 Fragments 创建占位符、视图重叠或为视图应用前景。

示例：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <ImageView  
        android:src="@drawable/nougat"  
        android:scaleType="fitCenter"  
        android:layout_height="match_parent"  
        android:layout_width="match_parent" />
```



Section 8.9: FrameLayout

[FrameLayout](#) is designed to block out an area on the screen to display a single item. You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the [android:layout_gravity](#) attribute.

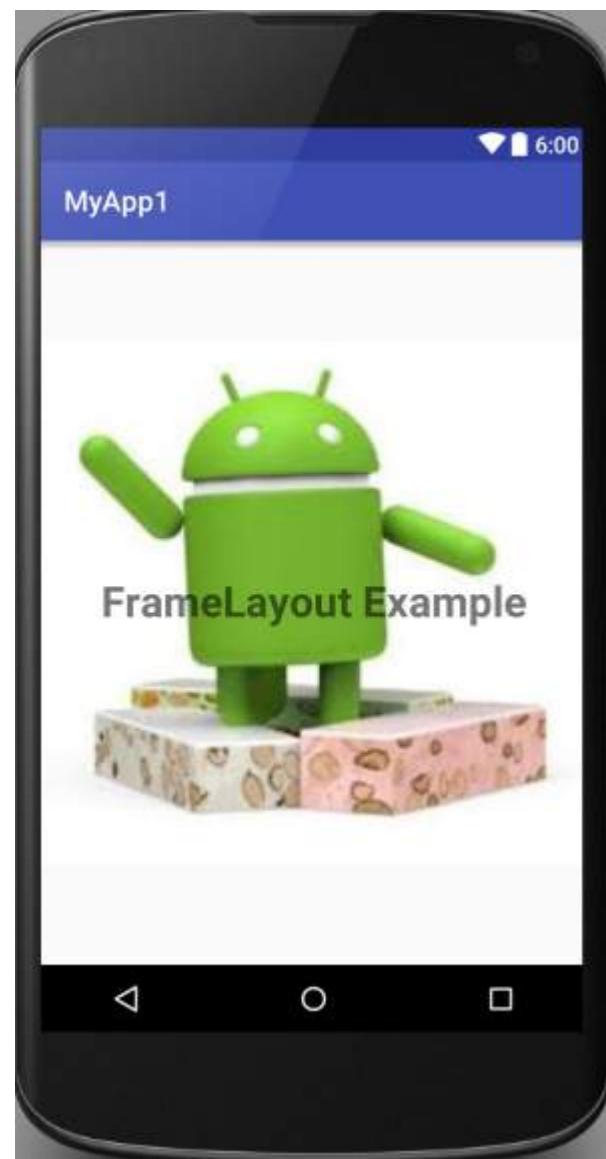
Generally, FrameLayout is used to hold a single child view. Common use cases are creating place holders for inflating Fragments in Activity, overlapping views or applying foreground to the views.

Example:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <ImageView  
        android:src="@drawable/nougat"  
        android:scaleType="fitCenter"  
        android:layout_height="match_parent"  
        android:layout_width="match_parent" />
```

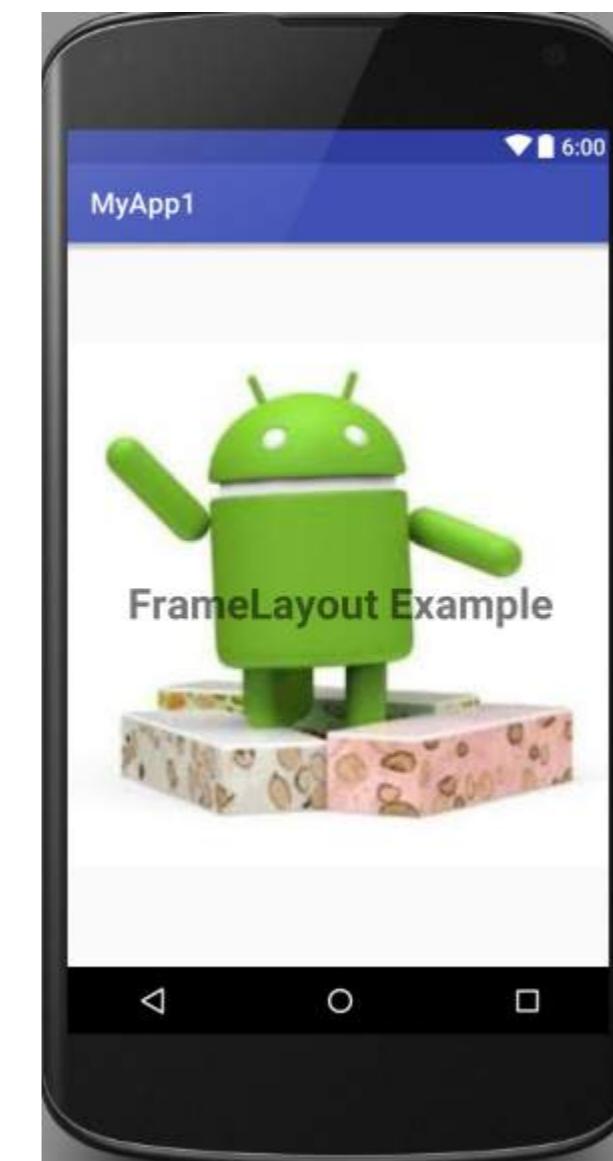
```
<TextView  
    android:text="FrameLayout 示例"  
    android:textSize="30sp"  
    android:textStyle="bold"  
    android:layout_height="match_parent"  
    android:layout_width="match_parent"  
    android:gravity="center" />  
  
</FrameLayout>
```

它将如下所示：



```
<TextView  
    android:text="FrameLayout Example"  
    android:textSize="30sp"  
    android:textStyle="bold"  
    android:layout_height="match_parent"  
    android:layout_width="match_parent"  
    android:gravity="center" />  
  
</FrameLayout>
```

It will look like this:



第8.10节：GridLayout

GridLayout，顾名思义，是一种用于将视图排列成网格的布局。GridLayout将自身划分为列和行。如下面的示例所示，列数和/或行数由属性columnCount和rowCount指定。向此布局添加视图时，第一个视图将添加到第一列，第二个视图添加到第二列，第三个视图添加到第二行的第一列。

```
<?xml version="1.0" encoding="utf-8"?>  
<GridLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"
```

Section 8.10: GridLayout

GridLayout, as the name suggests is a layout used to arrange Views in a grid. A GridLayout divides itself into columns and rows. As you can see in the example below, the amount of columns and/or rows is specified by the properties columnCount and rowCount. Adding Views to this layout will add the first view to the first column, the second view to the second column, and the third view to the first column of the second row.

```
<?xml version="1.0" encoding="utf-8"?>  
<GridLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"
```

```
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:columnCount="2"
        android:rowCount="2"

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

</GridLayout>
```

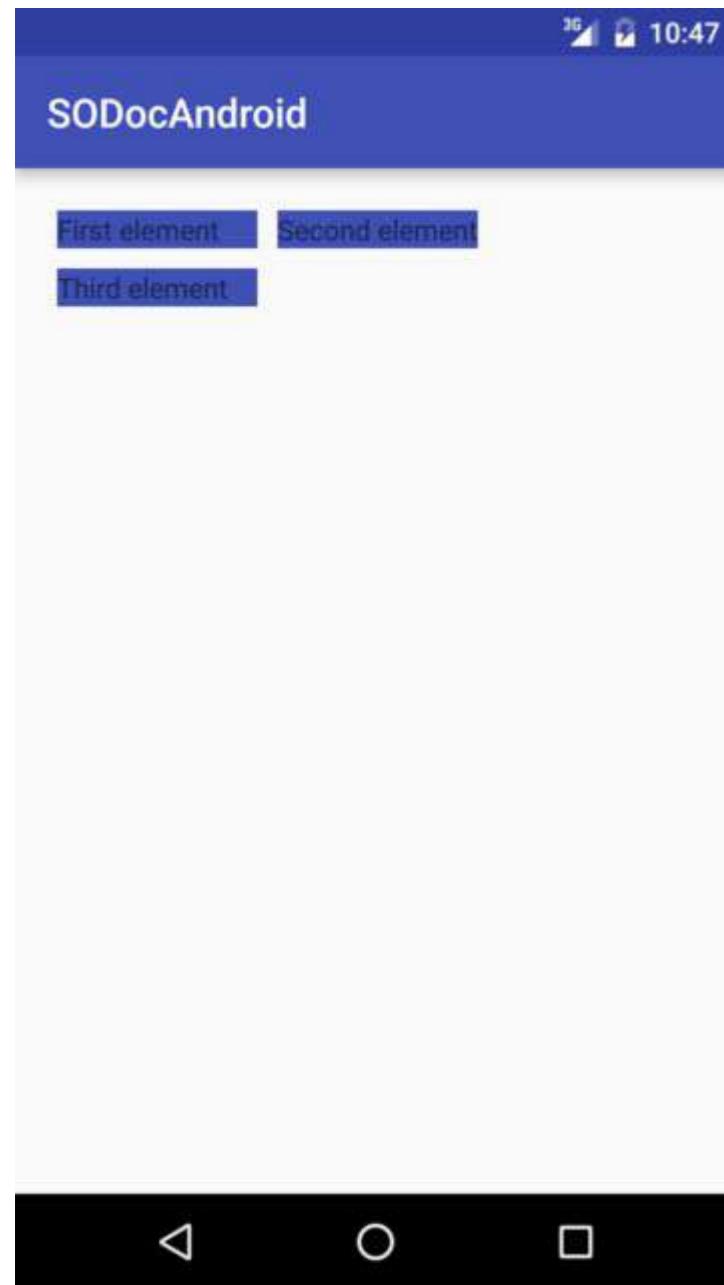
```
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:columnCount="2"
        android:rowCount="2"

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

</GridLayout>
```



第8.11节：CoordinatorLayout

版本 ≥ 2.3

CoordinatorLayout 是一个类似于 FrameLayout 的容器，但具有额外的功能，在官方文档中被称为超级强化的 FrameLayout。

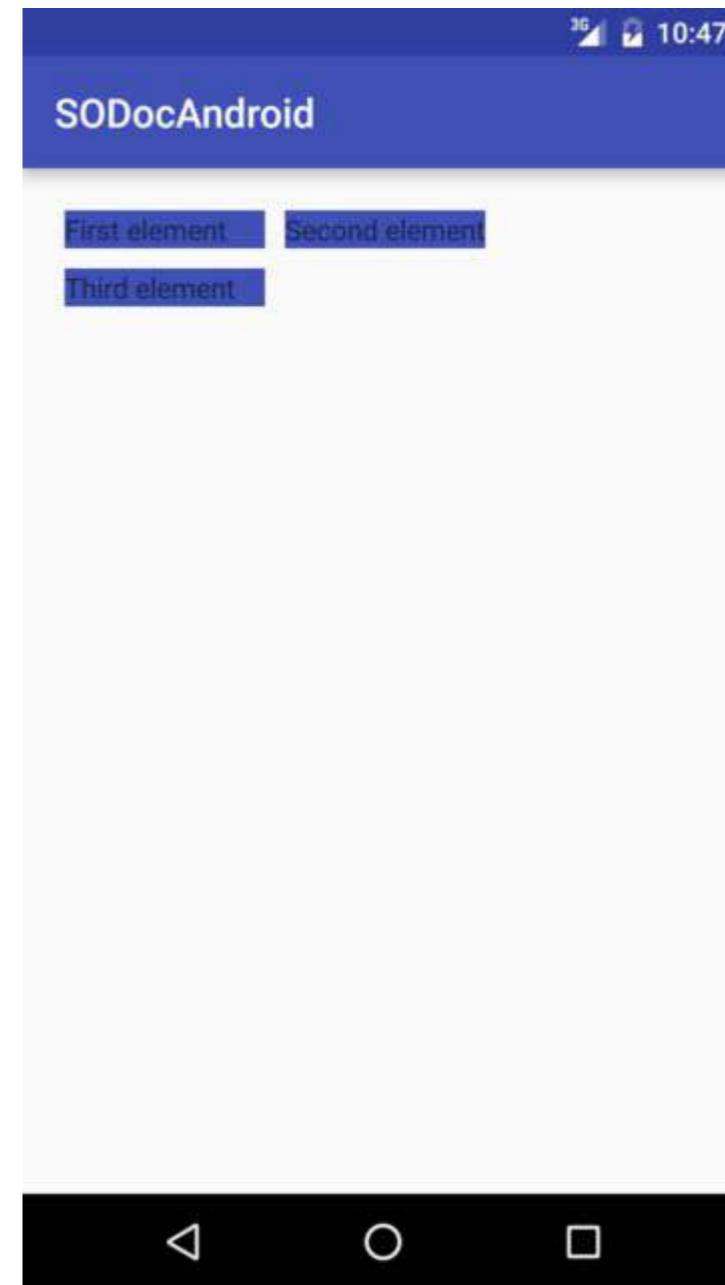
通过将 CoordinatorLayout.Behavior 附加到 CoordinatorLayout 的直接子视图，您将能够拦截触摸事件、窗口插入、测量、布局和嵌套滚动。

为了使用它，您首先需要在 gradle 文件中添加对支持库的依赖：

```
compile 'com.android.support:design:25.3.1'
```

该库最新版本号可在这里找到

CoordinatorLayout 的一个实际用例是创建带有 FloatingActionButton 的视图。在这个特定的案例中，我们将创建一个带有 SwipeRefreshLayout 和 FloatingActionButton 的 RecyclerView。以下是实现方法：



Section 8.11: CoordinatorLayout

Version ≥ 2.3

The CoordinatorLayout is a container somewhat similar to FrameLayout but with extra capabilities, it is called super-powered FrameLayout in the official documentation.

By attaching a [CoordinatorLayout.Behavior](#) to a direct child of CoordinatorLayout, you'll be able to intercept touch events, window insets, measurement, layout, and nested scrolling.

In order to use it, you will first have to add a dependency for the support library in your gradle file:

```
compile 'com.android.support:design:25.3.1'
```

The number of the latest version of the library may be found [here](#)

One practical use case of the CoordinatorLayout is creating a view with a FloatingActionButton. In this specific case, we will create a RecyclerView with a SwipeRefreshLayout and a FloatingActionButton on top of that. Here's how you can do that:

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coord_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/swipe_refresh_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/recycler_view"/>

    </android.support.v4.widget.SwipeRefreshLayout>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:clickable="true"
        android:color="@color/colorAccent"
        android:src="@mipmap/ic_add_white"
        android:layout_gravity="end|bottom"
        app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>

```

注意FloatingActionButton是如何锚定到CoordinatorLayout上的

app:layout_anchor="@+id/coord_layout"

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coord_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/swipe_refresh_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/recycler_view"/>

    </android.support.v4.widget.SwipeRefreshLayout>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:clickable="true"
        android:color="@color/colorAccent"
        android:src="@mipmap/ic_add_white"
        android:layout_gravity="end|bottom"
        app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>

```

Notice how the FloatingActionButton is anchored to the CoordinatorLayout with

app:layout_anchor="@+id/coord_layout"

第9章：ConstraintLayout

参数	详情
子项	要添加到布局的视图
索引	布局层次结构中View的索引
参数	View的LayoutParams
属性	定义LayoutParams的AttributeSet
视图	已添加或移除的View
已更改	指示该View的大小或位置是否已更改
左	相对于父视图的左侧位置
上	相对于父视图的顶部位置
右	相对于父视图的右侧位置
下	相对于父视图的底部位置
widthMeasureSpec	父视图施加的水平空间要求
heightMeasureSpec	父视图施加的垂直空间要求
layoutDirection	-
a	-
widthAttr	-
heightAttr	-

ConstraintLayout 是一个 ViewGroup，允许你以灵活的方式定位和调整控件的大小。它兼容 Android 2.3 (API 级别 9) 及更高版本。

它允许你创建大型且复杂的布局，同时保持扁平的视图层级。它类似于 RelativeLayout，因为所有视图都是根据兄弟视图和父布局之间的关系进行布局，但它比 RelativeLayout 更灵活，并且更容易与 Android Studio 的布局编辑器一起使用。

第9.1节：将 ConstraintLayout 添加到你的项目中

要使用 ConstraintLayout，你需要 Android Studio 版本 2.2 或更高，并且至少拥有版本 32 (或更高版本) 的 Android Support Repository。

- 在你的 build.gradle 文件中添加 Constraint Layout 库作为依赖：

```
dependencies {  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
}
```

- 同步项目

要向你的项目添加新的约束布局：

- 右键点击你的模块布局目录，然后点击新建>XML>布局XML。
- 输入布局的名称，并为根标签输入"android.support.constraint.ConstraintLayout"。
- 点击完成。

否则只需添加一个布局文件：

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"
```

Chapter 9: ConstraintLayout

Parameter	Details
child	The View to be added to the layout
index	The index of the View in the layout hierarchy
params	The LayoutParams of the View
attrs	The AttributeSet that defines the LayoutParams
view	The View that has been added or removed
changed	Indicates if this View has changed size or position
left	The left position, relative to the parent View
top	The top position, relative to the parent View
right	The right position, relative to the parent View
bottom	The bottom position, relative to the parent View
widthMeasureSpec	The horizontal space requirements imposed by the parent View
heightMeasureSpec	The vertical space requirements imposed by the parent View
layoutDirection	-
a	-
widthAttr	-
heightAttr	-

ConstraintLayout is a ViewGroup which allows you to position and size widgets in a flexible way. It is compatible with Android 2.3 (API level 9) and higher.

It allows you to create large and complex layouts with a flat view hierarchy. It is similar to RelativeLayout in that all views are laid out according to relationships between sibling views and the parent layout, but it's more flexible than RelativeLayout and easier to use with Android Studio's Layout Editor.

Section 9.1: Adding ConstraintLayout to your project

To work with ConstraintLayout, you need Android Studio Version 2.2 or newer and have at least version 32 (or higher) of Android Support Repository.

- Add the Constraint Layout library as a dependency in your build.gradle file:

```
dependencies {  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
}
```

- Sync project

To add a new constraint layout to your project:

- Right-click on your module's layout directory, then click New > XML > Layout XML.
- Enter a name for the layout and enter "android.support.constraint.ConstraintLayout" for the Root Tag.
- Click Finish.

Otherwise just add in a layout file:

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout>
```

第9.2节：链 (Chains)

自ConstraintLayout alpha 9版本起，链 (Chains) 功能可用。链是ConstraintLayout内一组视图，它们之间以双向方式连接，即A通过约束连接到B，且B通过另一个约束连接回A。

示例：

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- 此视图与 bottomTextView 关联 -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainPacked="true"/>

    <!-- 此视图同时链接到topTextView -->
    <TextView
        android:id="@+id/bottomTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom\nMkay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topTextView" />

</android.support.constraint.ConstraintLayout>
```

在此示例中，两个视图垂直排列，且都垂直居中。您可以通过调整链的bias来改变这些视图的垂直位置。将以下代码添加到链的第一个元素中：

```
app:layout_constraintVertical_bias="0.2"
```

在垂直链中，第一个元素是最顶部的视图；在水平链中，第一个元素是最左侧的视图。第一个元素定义整个链的行为。

链是一个新功能，且经常更新。这里是官方的Android链文档。[Here](#)

```
xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout>
```

Section 9.2: Chains

Since ConstraintLayout alpha 9, **Chains** are available. A **Chain** is a set of views inside a ConstraintLayout that are connected in a bi-directional way between them, i.e **A** connected to **B** with a constraint, and **B** connected to **A** with another constraint.

Example:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- this view is linked to the bottomTextView -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainPacked="true"/>

    <!-- this view is linked to the topTextView at the same time -->
    <TextView
        android:id="@+id/bottomTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom\nMkay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topTextView" />

</android.support.constraint.ConstraintLayout>
```

In this example, the two views are positioned one under another and both of them are centered vertically. You may change the vertical position of these views by adjusting the chain's **bias**. Add the following code to the first element of a chain:

```
app:layout_constraintVertical_bias="0.2"
```

In a vertical chain, the first element is a top-most view, and in a horizontal chain it is the left-most view. The first element defines the whole chain's behavior.

Chains are a new feature and are updated frequently. [Here](#) is an official Android Documentation on Chains.

第10章：TextInputLayout

TextInputLayout 是为在 EditText 上显示浮动标签而引入的。EditText 必须被 TextInputLayout 包裹，才能显示浮动标签。

第10.1节：基本用法

这是TextInputLayout的基本用法。

请确保按照备注部分的说明，在build.gradle文件中添加依赖。

示例：

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/username"/>

</android.support.design.widget.TextInputLayout>
```

第10.2节：密码可见性切换

使用输入密码类型时，您还可以启用一个图标，该图标可以显示或隐藏整个文本 [passwordToggleEnabled 属性](#)。

您也可以使用以下属性自定义相同的默认设置：

- [passwordToggleDrawable](#)：更改默认的眼睛图标
- [passwordToggleTint](#)：为密码可见性切换图标应用色调。
- [passwordToggleTintMode](#)：指定用于应用背景色调的混合模式。

示例：

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleContentDescription="@string/description"
    app:passwordToggleDrawable="@drawable/another_toggle_drawable"
    app:passwordToggleEnabled="true">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

第10.3节：添加字符计数

TextInputLayout 为其中定义的 EditText 提供了一个字符计数器。计数器将显示在 EditText 的下方。

只需使用[setCounterEnabled\(\)](#)和[setCounterMaxLength](#)方法：

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
```

Chapter 10: TextInputLayout

TextInputLayout was introduced to display the floating label on EditText. The EditText has to be wrapped by TextInputLayout in order to display the floating label.

Section 10.1: Basic usage

It is the basic usage of the TextInputLayout.

Make sure to add the dependency in the build.gradle file as described in the remarks section.

Example:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/username"/>

</android.support.design.widget.TextInputLayout>
```

Section 10.2: Password Visibility Toggles

With an input password type, you can also [enable an icon that can show or hide](#) the entire text using the [passwordToggleEnabled](#) attribute.

You can also customize same default using these attributes:

- [passwordToggleDrawable](#): to change the default eye icon
- [passwordToggleTint](#): to apply a tint to the password visibility toggle drawable.
- [passwordToggleTintMode](#): to specify the blending mode used to apply the background tint.

Example:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleContentDescription="@string/description"
    app:passwordToggleDrawable="@drawable/another_toggle_drawable"
    app:passwordToggleEnabled="true">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

Section 10.3: Adding Character Counting

The TextInputLayout has a [character counter](#) for an EditText defined within it. The counter will be rendered below the EditText.

Just use the [setCounterEnabled\(\)](#) and [setCounterMaxLength](#) methods:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
```

```
til.setCounterEnabled(true);  
til.setCounterMaxLength(15);
```

或者在xml中使用app:counterEnabled和app:counterMaxLength属性。

```
<android.support.design.widget.TextInputLayout  
    app:counterEnabled="true"  
    app:counterMaxLength="15">  
  
    <EditText/>  
  
</android.support.design.widget.TextInputLayout>
```

第10.4节：错误处理

你可以使用TextInputLayout根据Material Design指南通过[setError](#)和[setErrorEnabled](#)方法显示错误信息。

为了在EditText下方显示错误，使用：

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);  
til.setErrorEnabled(true);  
til.setError("你需要输入一个名字");
```

要启用TextInputLayout中的错误提示，可以在xml中使用app:errorEnabled="true"，或者如上所示使用[til.setErrorEnabled\(true\)](#)。

您将获得：



```
til.setCounterEnabled(true);  
til.setCounterMaxLength(15);
```

or the [app:counterEnabled](#) and [app:counterMaxLength](#) attributes in the xml.

```
<android.support.design.widget.TextInputLayout  
    app:counterEnabled="true"  
    app:counterMaxLength="15">  
  
    <EditText/>  
  
</android.support.design.widget.TextInputLayout>
```

Section 10.4: Handling Errors

You can use the TextInputLayout to display error messages according to the [material design guidelines](#) using the [setError](#) and [setErrorEnabled](#) methods.

In order to show the error below the EditText use:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);  
til.setErrorEnabled(true);  
til.setError("You need to enter a name");
```

To enable error in the TextInputLayout you can either use [app:errorEnabled="true"](#) in xml or [til.setErrorEnabled\(true\)](#); as shown above.

You will obtain:



第10.5节：自定义TextInputLayout的外观

您可以通过在styles.xml中定义自定义样式，来自定义TextInputLayout及其嵌入的EditText的外观。定义的样式可以作为样式或主题添加到您的TextInputLayout中。

自定义提示文本外观的示例：

styles.xml：

```
<!--浮动标签文本样式-->  
<style name="MyHintStyle" parent="TextAppearance.AppCompat.Small">  
    <item name="android:textColor">@color/black</item>  
</style>  
  
<!--输入框样式-->  
<style name="MyEditText" parent="Theme.AppCompat.Light">  
    <item name="colorControlNormal">@color/indigo</item>  
    <item name="colorControlActivated">@color/pink</item>  
</style>
```

Section 10.5: Customizing the appearance of the TextInputLayout

You can customize the appearance of the TextInputLayout and its embedded EditText by defining custom styles in your [styles.xml](#). The defined styles can either be added as styles or themes to your TextInputLayout.

Example for customizing the hint appearance:

styles.xml:

```
<!--Floating label text style-->  
<style name="MyHintStyle" parent="TextAppearance.AppCompat.Small">  
    <item name="android:textColor">@color/black</item>  
</style>  
  
<!--Input field style-->  
<style name="MyEditText" parent="Theme.AppCompat.Light">  
    <item name="colorControlNormal">@color/indigo</item>  
    <item name="colorControlActivated">@color/pink</item>  
</style>
```

要应用样式，请按如下方式更新您的TextInputLayout和EditText

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:hintTextAppearance="@style/MyHintStyle">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string>Title"
        android:theme="@style/MyEditText" />

</android.support.design.widget.TextInputLayout>
```

自定义TextInputLayout强调色的示例。强调色会影响EditText基线的颜色以及浮动提示文本的颜色：

styles.xml：

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

布局文件：

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/textInputLayout_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/TextInputLayoutWithPrimaryColor">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/textInputEditText_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/login_hint_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

第10.6节：TextInputEditText

TextInputEditText 是一个带有额外修复的 EditText，用于在“提取”模式下在输入法编辑器（IME）中显示提示。

提取模式是当空间太小（例如智能手机横屏时）点击 EditText 时，键盘编辑器切换到的模式。

在这种情况下，使用 EditText 编辑文本时，你会发现输入法编辑器（IME）不会给出你正在编辑内容的提示。

TextInputEditText 解决了这个问题，在用户设备的输入法编辑器处于提取模式时提供提示文本。

示例：

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Description"
```

To Apply Style update your TextInputLayout And EditText as follows

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:hintTextAppearance="@style/MyHintStyle">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string>Title"
        android:theme="@style/MyEditText" />

</android.support.design.widget.TextInputLayout>
```

Example to customize the accent color of the TextInputLayout. The accent color affects the color of the baseline of the EditText and the text color for the floating hint text:

styles.xml:

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

layout file:

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/textInputLayout_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/TextInputLayoutWithPrimaryColor">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/textInputEditText_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/login_hint_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

Section 10.6: TextInputEditText

The [TextInputEditText](#) is an EditText with an extra fix to display a hint in the IME when in '[extract mode](#)'.

The **Extract mode** is the mode that the keyboard editor switches to when you click on an EditText when the space is too small (for example landscape on a smartphone).

In this case, using an EditText while you are editing the text you can see that the IME doesn't give you a hint of what you're editing

The TextInputEditText fixes this issue providing hint text while the user's device's IME is in Extract mode.

Example:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Description"
```

```
>
<android.support.design.widget.TextInputEditText
    android:id="@+id/description"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

</android.support.design.widget.TextInputLayout>
```

```
>
<android.support.design.widget.TextInputEditText
    android:id="@+id/description"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

</android.support.design.widget.TextInputLayout>
```

第11章：CoordinatorLayout和Behaviors

CoordinatorLayout是一个功能强大的FrameLayout，这个ViewGroup的目标是协调其中的视图。

CoordinatorLayout的主要吸引力在于它能够协调XML文件中视图的动画和过渡效果。

CoordinatorLayout主要用于两种场景：

:作为顶层应用装饰或界面布局

:作为与一个或多个子视图进行特定交互的容器

第11.1节：创建一个简单的Behavior

要创建一个Behavior，只需继承CoordinatorLayout.Behavior类。

继承CoordinatorLayout.Behavior

示例：

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {  
  
    /**  
     * 默认构造函数。  
     */  
    public MyBehavior() {  
    }  
  
    /**  
     * 用于从布局中膨胀 MyBehavior 的默认构造函数。  
     *  
     * @param context {@link Context}。  
     * @param attrs {@link AttributeSet}。  
     */  
    public MyBehavior(Context context, AttributeSet attrs) {  
        super(context, attrs);  
    }  
}
```

此行为需要附加到CoordinatorLayout的子视图上才能被调用。

以编程方式附加 Behavior

```
MyBehavior myBehavior = new MyBehavior();  
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) view.getLayoutParams();  
params.setBehavior(myBehavior);
```

在 XML 中附加 Behavior

您可以使用 layout_behavior 属性在 XML 中附加 Behavior：

```
<View  
    android:layout_height="...."  
    android:layout_width="...."
```

Chapter 11: CoordinatorLayout and Behaviors

The CoordinatorLayout is a super-powered FrameLayout and goal of this ViewGroup is to coordinate the views that are inside it.

The main appeal of the CoordinatorLayout is its ability to coordinate the animations and transitions of the views within the XML file itself.

CoordinatorLayout is intended for two primary use cases:

:As a top-level application decor or chrome layout

:As a container for a specific interaction with one or more child views

Section 11.1: Creating a simple Behavior

To create a Behavior just extend the [CoordinatorLayout.Behavior](#) class.

Extend the CoordinatorLayout.Behavior

Example:

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {  
  
    /**  
     * Default constructor.  
     */  
    public MyBehavior() {  
    }  
  
    /**  
     * Default constructor for inflating a MyBehavior from layout.  
     *  
     * @param context The {@link Context}.  
     * @param attrs The {@link AttributeSet}.  
     */  
    public MyBehavior(Context context, AttributeSet attrs) {  
        super(context, attrs);  
    }  
}
```

This behavior need to be attached to a child View of a CoordinatorLayout to be called.

Attach a Behavior programmatically

```
MyBehavior myBehavior = new MyBehavior();  
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) view.getLayoutParams();  
params.setBehavior(myBehavior);
```

Attach a Behavior in XML

You can use the layout_behavior attribute to attach the behavior in XML:

```
<View  
    android:layout_height="...."  
    android:layout_width="...."
```

```
app:layout_behavior=".MyBehavior" />
```

自动附加 Behavior

如果您正在使用自定义视图，可以使用以下方式附加 Behavior @CoordinatorLayout.DefaultBehavior 注解：

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {
}
```

第11.2节：使用SwipeDismissBehavior

SwipeDismissBehavior适用于任何视图，并在带有CoordinatorLayout的布局中实现滑动以关闭的功能。

只需使用：

```
final SwipeDismissBehavior<MyView> swipe = new SwipeDismissBehavior();

//设置此行为的滑动方向。
swipe.setSwipeDirection(
    SwipeDismissBehavior.SWIPE_DIRECTION_ANY);

//设置在关闭事件发生时使用的监听器
swipe.setListener(
    new SwipeDismissBehavior.OnDismissListener() {
        @Override public void onDismiss(View view) {
            //.....
        }
        @Override
        public void onDragStateChanged(int state) {
            //.....
        }
    });
};

//将 SwipeDismissBehavior 附加到视图
LayoutParams coordinatorParams =
    (LayoutParams) mView.getLayoutParams();
coordinatorParams.setBehavior(swipe);
```

第11.3节：创建视图之间的依赖关系

您可以使用CoordinatorLayout.Behavior来创建视图之间的依赖关系。您可以通过以下方式将一个View锚定到另一个View：

- 使用layout_anchor属性。
- 创建自定义的Behavior并实现返回true的layoutDependsOn方法。

例如，为了创建一个Behavior，使得当另一个ImageView（例如工具栏）移动时，该ImageView也随之移动，请执行以下步骤：

- 创建自定义的Behavior：

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
```

```
app:layout_behavior=".MyBehavior" />
```

Attach a Behavior automatically

If you are working with a custom view you can attach the behavior using the @CoordinatorLayout.DefaultBehavior annotation:

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {
}
```

Section 11.2: Using the SwipeDismissBehavior

The [SwipeDismissBehavior](#) works on any View and implements the functionality of swipe to dismiss in our layouts with a CoordinatorLayout.

Just use:

```
final SwipeDismissBehavior<MyView> swipe = new SwipeDismissBehavior();

//Sets the swipe direction for this behavior.
swipe.setSwipeDirection(
    SwipeDismissBehavior.SWIPE_DIRECTION_ANY);

//Set the listener to be used when a dismiss event occurs
swipe.setListener(
    new SwipeDismissBehavior.OnDismissListener() {
        @Override public void onDismiss(View view) {
            //.....
        }
        @Override
        public void onDragStateChanged(int state) {
            //.....
        }
    });

//Attach the SwipeDismissBehavior to a view
LayoutParams coordinatorParams =
    (LayoutParams) mView.getLayoutParams();
coordinatorParams.setBehavior(swipe);
```

Section 11.3: Create dependencies between Views

You can use the [CoordinatorLayout.Behavior](#) to create dependencies between views. You can anchor a [View](#) to another [View](#) by:

- using the [layout_anchor](#) attribute.
- creating a custom Behavior and implementing the [layoutDependsOn](#) method returning [true](#).

For example, in order to create a Behavior for moving an ImageView when another one is moved (example Toolbar), perform the following steps:

- Create the custom Behavior:

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
```

- 重写layoutDependsOn方法，返回true。每当布局发生变化时都会调用此方法：

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent,
    ImageView child, View dependency) {
    // 返回true以添加依赖关系。
    return dependency instanceof Toolbar;
}
```

- 每当layoutDependsOn方法返回true时，都会调用onDependentViewChanged方法：

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, ImageView child, View
dependency) {
    // 在这里实现动画、平移或移动；始终与提供的依赖项相关。

    float translationY = Math.min(0, dependency.getTranslationY() - dependency.getHeight());
    child.setTranslationY(translationY);
}
```

- Override the `layoutDependsOn` method returning `true`. This method is called every time a change occurs to the layout:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent,
    ImageView child, View dependency) {
    // Returns true to add a dependency.
    return dependency instanceof Toolbar;
}
```

- Whenever the method `layoutDependsOn` returns `true` the method `onDependentViewChanged` is called:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, ImageView child, View
dependency) {
    // Implement here animations, translations, or movements; always related to the provided
    // dependency.
    float translationY = Math.min(0, dependency.getTranslationY() - dependency.getHeight());
    child.setTranslationY(translationY);
}
```

第12章：TabLayout

第12.1节：在没有ViewPager的情况下使用TabLayout

大多数情况下，TabLayout 会与ViewPager一起使用，以获得随之而来的滑动功能。

可以通过使用TabLayout.OnTabSelectedListener来在没有ViewPager的情况下使用TabLayout。

首先，在你的活动的XML文件中添加一个TabLayout：

```
<android.support.design.widget.TabLayout  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:id="@+id/tabLayout" />
```

对于在Activity内的导航，根据选中的标签手动填充UI。

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);  
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
    @Override  
    public void onTabSelected(TabLayout.Tab tab) {  
        int position = tab.getPosition();  
        switch (tab.getPosition()) {  
            case 1:  
                getSupportFragmentManager().beginTransaction()  
                    .replace(R.id.fragment_container, new ChildFragment()).commit();  
                break;  
            // 继续处理 TabLayout 中的每个标签页  
        }  
  
        @Override  
        public void onTabUnselected(TabLayout.Tab tab) {  
  
        }  
  
        @Override  
        public void onTabReselected(TabLayout.Tab tab) {  
    }  
});
```

Chapter 12: TabLayout

Section 12.1: Using a TabLayout without a ViewPager

Most of the time a TabLayout is used together with a ViewPager, in order to get the swipe functionality that comes with it.

It is possible to use a TabLayout without a ViewPager by using a TabLayout.OnTabSelectedListener.

First, add a TabLayout to your activity's XML file:

```
<android.support.design.widget.TabLayout  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:id="@+id/tabLayout" />
```

For navigation within an Activity, manually populate the UI based on the tab selected.

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);  
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
    @Override  
    public void onTabSelected(TabLayout.Tab tab) {  
        int position = tab.getPosition();  
        switch (tab.getPosition()) {  
            case 1:  
                getSupportFragmentManager().beginTransaction()  
                    .replace(R.id.fragment_container, new ChildFragment()).commit();  
                break;  
            // Continue for each tab in TabLayout  
        }  
  
        @Override  
        public void onTabUnselected(TabLayout.Tab tab) {  
  
        }  
  
        @Override  
        public void onTabReselected(TabLayout.Tab tab) {  
    }  
});
```

第13章：ViewPager

ViewPager 是一个布局管理器，允许用户左右滑动浏览数据页面。它通常与 Fragment 一起使用，Fragment 是一种方便的方式来提供和管理每个页面的生命周期。

第13.1节：带点指示器的 ViewPager



我们需要的全部是：ViewPager、TabLayout 和两个用于选中与默认状态的点的 drawable。

首先，我们必须将TabLayout添加到屏幕布局中，并将其与ViewPager连接。我们可以通过两种方式实现：

嵌套在 ViewPager 中的 TabLayout

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

在这种情况下，TabLayout 会自动与 ViewPager 连接，但 TabLayout 会位于 ViewPager 旁边，而不是覆盖在它上面。

独立的 TabLayout

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Chapter 13: ViewPager

ViewPager is a Layout manager that allows the user to flip left and right through pages of data. It is most often used in conjunction with Fragment, which is a convenient way to supply and manage the lifecycle of each page.

Section 13.1: ViewPager with a dots indicator



All we need are: [ViewPager](#), [TabLayout](#) and 2 drawables for selected and default dots.

Firstly, we have to add TabLayout to our screen layout, and connect it with ViewPager. We can do this in two ways:

Nested TabLayout in ViewPager

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

In this case TabLayout will be automatically connected with ViewPager, but TabLayout will be next to ViewPager, not over him.

Separate TabLayout

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tab_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

在这种情况下，我们可以将 TabLayout 放在任何位置，但必须通过编程方式将 TabLayout 与 ViewPager 连接起来

```
ViewPager pager = (ViewPager) view.findViewById(R.id.photos_viewpager);  
PagerAdapter adapter = new PhotosAdaptergetChildFragmentManager(), photosUrl);  
pager.setAdapter(adapter);  
  
TabLayout tabLayout = (TabLayout) view.findViewById(R.id.tab_layout);  
tabLayout.setupWithViewPager(pager, true);
```

一旦我们创建了布局，就必须准备我们的点。因此，我们创建了三个文件：selected_dot.xml、default_dot.xml 和 tab_selector.xml。

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">  
    <item>  
        <shape  
            android:innerRadius="0dp"  
            android:shape="ring"  
            android:thickness="8dp"  
            android:useLevel="false">  
            <solid android:color="@color/colorAccent"/>  
        </shape>  
    </item>  
</layer-list>
```

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">  
    <item>  
        <shape  
            android:innerRadius="0dp"  
            android:shape="ring"  
            android:thickness="8dp"  
            android:useLevel="false">  
            <solid android:color="@android:color/darker_gray"/>  
        </shape>  
    </item>  
</layer-list>
```

tab_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
  
    <item android:drawable="@drawable/selected_dot"  
        android:state_selected="true"/>  
  
    <item android:drawable="@drawable/default_dot"/>
```

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tab_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

In this case, we can put TabLayout anywhere, but we have to connect TabLayout with ViewPager programmatically

```
ViewPager pager = (ViewPager) view.findViewById(R.id.photos_viewpager);  
PagerAdapter adapter = new PhotosAdaptergetChildFragmentManager(), photosUrl);  
pager.setAdapter(adapter);  
  
TabLayout tabLayout = (TabLayout) view.findViewById(R.id.tab_layout);  
tabLayout.setupWithViewPager(pager, true);
```

Once we created our layout, we have to prepare our dots. So we create three files: selected_dot.xml, default_dot.xml and tab_selector.xml.

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">  
    <item>  
        <shape  
            android:innerRadius="0dp"  
            android:shape="ring"  
            android:thickness="8dp"  
            android:useLevel="false">  
            <solid android:color="@color/colorAccent"/>  
        </shape>  
    </item>  
</layer-list>
```

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">  
    <item>  
        <shape  
            android:innerRadius="0dp"  
            android:shape="ring"  
            android:thickness="8dp"  
            android:useLevel="false">  
            <solid android:color="@android:color/darker_gray"/>  
        </shape>  
    </item>  
</layer-list>
```

tab_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
  
    <item android:drawable="@drawable/selected_dot"  
        android:state_selected="true"/>  
  
    <item android:drawable="@drawable/default_dot"/>
```

```
</selector>
```

现在我们只需在xml布局中的TabLayout添加3行代码，就完成了。

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

第13.2节：带有碎片的基本ViewPager用法

ViewPager允许在一个活动中显示多个碎片，可以通过向左或向右滑动进行导航。

ViewPager需要通过PagerAdapter来提供视图或碎片。

然而，在使用碎片（Fragments）的情况下，你会发现两个更具体的实现最为有用，分别是FragmentPagerAdapter和FragmentStatePagerAdapter。当碎片首次需要实例化时，`getItem(position)`会被调用以实例化每个位置的碎片。`getCount()`方法会返回页面的总数，以便ViewPager知道需要显示多少个碎片。

FragmentPagerAdapter和FragmentStatePagerAdapter都会缓存ViewPager需要显示的Fragment。默认情况下，ViewPager会尝试存储最多3个Fragment，分别对应当前可见的Fragment以及左右相邻的Fragment。同时，FragmentStatePagerAdapter会保存每个Fragment的状态。

请注意，这两种实现都假设你的Fragment会保持其位置，因此如果你维护的是Fragment列表，而不是像`getItem()`方法中那样拥有固定数量的Fragment，你需要创建一个PagerAdapter的子类，并至少重写`instantiateItem()`、`destroyItem()`和`getItemPosition()`方法。

只需按照基本示例在布局中添加一个ViewPager：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

然后定义适配器，用于确定存在多少页面以及每个页面显示哪个Fragment。

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        //应用适配器
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPage = (ViewPager) findViewById(R.id.view_pager);
        mViewPage.setAdapter(mFragmentAdapter);
    }
}
```

```
</selector>
```

Now we need to add only 3 lines of code to TabLayout in our xml layout and you're done.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Section 13.2: Basic ViewPager usage with fragments

A ViewPager allows to show multiple fragments in an activity that can be navigated by either flipping left or right. A ViewPager needs to be feed of either Views or Fragments by using a PagerAdapter.

There are however two more specific implementations that you will find most useful in case of using Fragments which are FragmentPagerAdapter and FragmentStatePagerAdapter. When a Fragment needs to be instantiated for the first time, `getItem(position)` will be called for each position that needs instantiating. The `getCount()` method will return the total number of pages so the ViewPager knows how many Fragments need to be shown.

Both FragmentPagerAdapter and FragmentStatePagerAdapter keep a cache of the Fragments that the ViewPager will need to show. By default the ViewPager will try to store a maximum of 3 Fragments that correspond to the currently visible Fragment, and the ones next to the right and left. Also FragmentStatePagerAdapter will keep the state of each of your fragments.

Be aware that both implementations assume your fragments will keep their positions, so if you keep a list of the fragments instead of having a static number of them as you can see in the `getItem()` method, you will need to create a subclass of PagerAdapter and override at least `instantiateItem()`, `destroyItem()` and `getItemPosition()` methods.

Just add a ViewPager in your layout as described in the basic example:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Then define the adapter that will determine how many pages exist and which fragment to display for each page of the adapter.

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        //Apply the Adapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPage = (ViewPager) findViewById(R.id.view_pager);
        mViewPage.setAdapter(mFragmentAdapter);
    }
}
```

```

private class MyPagerAdapter extends FragmentPagerAdapter{

    public MyPagerAdapter(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    // 返回该页面要显示的片段
    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new Fragment1();

            case 1:
                return new Fragment2();

            case 2:
                return new Fragment3();

            default:
                return null;
        }
    }

    // 返回总页数
    @Override
    public int getCount() {
        return 3;
    }
}

```

版本 ≥ 3.2.x

如果你使用的是 android.app.Fragment，你需要添加此依赖：

```
compile 'com.android.support:support-v13:25.3.1'
```

如果你使用 android.support.v4.app.Fragment，你必须添加这个依赖：

```
compile 'com.android.support:support-fragment:25.3.1'
```

第13.3节：带有PreferenceFragment的ViewPager

直到最近，使用 android.support.v4.app.FragmentPagerAdapter 会阻止将 PreferenceFragment 作为FragmentPagerAdapter中使用的Fragment之一。

最新版本的support v7库现在包含了 [PreferenceFragmentCompat](#) 类，它可以与 ViewPager和v4版本的FragmentPagerAdapter一起使用。

继承自 PreferenceFragmentCompat 的示例Fragment：

```

import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {

```

```

private class MyPagerAdapter extends FragmentPagerAdapter{

    public MyPagerAdapter(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    // Returns the fragment to display for that page
    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new Fragment1();

            case 1:
                return new Fragment2();

            case 2:
                return new Fragment3();

            default:
                return null;
        }
    }

    // Returns total number of pages
    @Override
    public int getCount() {
        return 3;
    }
}

```

Version ≥ 3.2.x

If you are using android.app.Fragment you have to add this dependency:

```
compile 'com.android.support:support-v13:25.3.1'
```

If you are using android.support.v4.app.Fragment you have to add this dependency:

```
compile 'com.android.support:support-fragment:25.3.1'
```

Section 13.3: ViewPager with PreferenceFragment

Until recently, using android.support.v4.app.FragmentPagerAdapter would prevent the usage of a PreferenceFragment as one of the Fragments used in the FragmentPagerAdapter.

The latest versions of the support v7 library now include the [PreferenceFragmentCompat](#) class, which will work with a ViewPager and the v4 version of FragmentPagerAdapter.

Example Fragment that extends PreferenceFragmentCompat:

```

import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {

```

```

    // 必需的空公共构造函数
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.fragment_settings_pref);
}

@Override
public void onCreatePreferences(Bundle bundle, String s) {
}
}

```

你现在可以在一个`android.support.v4.app.FragmentPagerAdapter`子类中使用这个Fragment：

```

private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new FragmentOne();

            case 1:
                return new FragmentTwo();

            case 2:
                return new MySettingsPrefFragment();

            default:
                return null;
        }
    }

    // .....
}

```

第13.4节：添加ViewPager

确保在应用的`build.gradle`文件的`dependencies`部分添加以下依赖：

```
compile 'com.android.support:support-core-ui:25.3.0'
```

然后将ViewPager添加到你的活动布局中：

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

然后定义你的PagerAdapter：

```

    // Required empty public constructor
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.fragment_settings_pref);
}

@Override
public void onCreatePreferences(Bundle bundle, String s) {
}
}

```

You can now use this Fragment in a `android.support.v4.app.FragmentPagerAdapter` subclass:

```

private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new FragmentOne();

            case 1:
                return new FragmentTwo();

            case 2:
                return new MySettingsPrefFragment();

            default:
                return null;
        }
    }

    // .....
}

```

Section 13.4: Adding a ViewPager

Make sure the following dependency is added to your app's `build.gradle` file under dependencies:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Then add the ViewPager to your activity layout:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

Then define your [PagerAdapter](#):

```

public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {

        // 为给定位置创建页面。例如：
        LayoutInflator inflater = LayoutInflator.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection, false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
        // 移除指定位置的页面。例如：
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        // 返回可用视图的数量。
        return numberOfPages;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        // 判断页面视图是否与instantiateItem(ViewGroup, int)返回的特定键对象相关联。例如：
        return view == object;
    }
}

```

最后在你的Activity中设置ViewPager：

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}

```

第13.5节：设置OnPageChangeListener

如果需要监听所选页面的变化，可以在ViewPager上实现[ViewPager.OnPageChangeListener](#)监听器：

```

viewPager.addOnPageChangeListener(new OnPageChangeListener() {

    // 当新页面被选中时将调用此方法。动画不一定
}

```

```

public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {

        // Create the page for the given position. For example:
        LayoutInflator inflater = LayoutInflator.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection, false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
        // Remove a page for the given position. For example:
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        //Return the number of views available.
        return numberOfPages;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        // Determines whether a page View is associated with a specific key object
        // as returned by instantiateItem(ViewGroup, int). For example:
        return view == object;
    }
}

```

Finally setup the ViewPager in your Activity:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}

```

Section 13.5: Setup OnPageChangeListener

If you need to listen for changes to the page selected you can implement the [ViewPager.OnPageChangeListener](#) listener on the ViewPager:

```

viewPager.addOnPageChangeListener(new OnPageChangeListener() {

    // This method will be invoked when a new page becomes selected. Animation is not necessarily
}

```

完成。

```
@Override  
public void onPageSelected(int position) {  
    // 你的代码  
}  
  
// 当当前页面被滚动时调用此方法，无论是程序发起的平滑滚动还是用户发起的触摸滚动。  
  
@Override  
public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
    // 你的代码  
}  
  
// 当滚动状态改变时调用。用于检测用户何时开始拖动，何时页面自动稳定到当前页，  
// 或者何时完全停止/空闲。  
  
@Override  
public void onPageScrollStateChanged(int state) {  
    // 你的代码  
}  
});
```

第13.6节：带有TabLayout的ViewPager

可以使用TabLayout来实现更便捷的导航。

你可以通过使用TabLayout.newTab()方法为适配器中的每个片段设置标签，但还有另一种更方便、更简单的方法，即TabLayout.setupWithViewPager()。

该方法会根据与你的

ViewPager关联的适配器内容，每次调用时同步创建和移除标签。

此外，它还会设置一个回调，使得每次用户翻页时，相应的标签会被选中。

只需定义一个布局

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout>  
  
<android.support.design.widget.TabLayout  
    android:id="@+id/tabs"  
    app:tabMode="scrollable" />  
  
<android.support.v4.view.ViewPager  
    android:id="@+id/viewpager"  
    android:layout_width="match_parent"  
    android:layout_height="0px"  
    android:layout_weight="1" />  
  
</LinearLayout>
```

然后实现FragmentPagerAdapter并将其应用于ViewPager：

```
public class MyViewPagerActivity extends AppCompatActivity {  
    private static final String TAG = MyViewPagerActivity.class.getName();  
  
    private MyPagerAdapter mFragmentAdapter;  
    private ViewPager mViewPage;  
    private TabLayout mTabLayout;  
  
    @Override
```

complete.

```
@Override  
public void onPageSelected(int position) {  
    // Your code  
}  
  
// This method will be invoked when the current page is scrolled, either as part of  
// a programmatically initiated smooth scroll or a user initiated touch scroll.  
@Override  
public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
    // Your code  
}  
  
// Called when the scroll state changes. Useful for discovering when the user begins  
// dragging, when the pager is automatically settling to the current page,  
// or when it is fully stopped/idle.  
@Override  
public void onPageScrollStateChanged(int state) {  
    // Your code  
}  
});
```

Section 13.6: ViewPager with TabLayout

A TabLayout can be used for easier navigation.

You can set the tabs for each fragment in your adapter by using TabLayout.newTab() method but there is another more convenient and easier method for this task which is [TabLayout.setupWithViewPager\(\)](#).

This method will sync by creating and removing tabs according to the contents of the adapter associated with your ViewPager each time you call it.

Also, it will set a callback so each time the user flips the page, the corresponding tab will be selected.

Just define a layout

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout>  
  
<android.support.design.widget.TabLayout  
    android:id="@+id/tabs"  
    app:tabMode="scrollable" />  
  
<android.support.v4.view.ViewPager  
    android:id="@+id/viewpager"  
    android:layout_width="match_parent"  
    android:layout_height="0px"  
    android:layout_weight="1" />  
  
</LinearLayout>
```

Then implement the FragmentPagerAdapter and apply it to the ViewPager:

```
public class MyViewPagerActivity extends AppCompatActivity {  
    private static final String TAG = MyViewPagerActivity.class.getName();  
  
    private MyPagerAdapter mFragmentAdapter;  
    private ViewPager mViewPage;  
    private TabLayout mTabLayout;  
  
    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.myActivityLayout);

    // 获取ViewPager并应用PagerAdapter
    mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.view_pager);
    mViewPager.setAdapter(mFragmentAdapter);

    // 将tabLayout和viewpager关联起来
    mTabLayout = (TabLayout) findViewById(R.id.tab_layout);
    mTabLayout.setupWithViewPager(mViewPager);
}

private class MyPagerAdapter extends FragmentPagerAdapter{

    public MyPagerAdapter(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    // 返回该页面要显示的fragment
    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new Fragment1();

            case 1:
                return new Fragment2();

            case 2:
                return new Fragment3();

            default:
                return null;
        }
    }

    // 将作为标签的标题显示
    @Override
    public CharSequence getPageTitle(int position) {
        switch(position) {
            case 0:
                return "Fragment 1 title";

            case 1:
                return "片段2标题";

            case 2:
                return "片段3标题";

            default:
                return null;
        }
    }

    // 返回总页数
    @Override
    public int getCount() {
        return 3;
    }
}

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.myActivityLayout);

    // Get the ViewPager and apply the PagerAdapter
    mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.view_pager);
    mViewPager.setAdapter(mFragmentAdapter);

    // link the tabLayout and the viewPager together
    mTabLayout = (TabLayout) findViewById(R.id.tab_layout);
    mTabLayout.setupWithViewPager(mViewPager);
}

private class MyPagerAdapter extends FragmentPagerAdapter{

    public MyPagerAdapter(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    // Returns the fragment to display for that page
    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new Fragment1();

            case 1:
                return new Fragment2();

            case 2:
                return new Fragment3();

            default:
                return null;
        }
    }

    // Will be displayed as the tab's label
    @Override
    public CharSequence getPageTitle(int position) {
        switch(position) {
            case 0:
                return "Fragment 1 title";

            case 1:
                return "Fragment 2 title";

            case 2:
                return "Fragment 3 title";

            default:
                return null;
        }
    }

    // Returns total number of pages
    @Override
    public int getCount() {
        return 3;
    }
}

```

}

}

第14章：CardView

参数	详情
cardBackgroundColor	CardView的背景颜色。
cardCornerRadius	CardView的圆角半径。
cardElevation	CardView 的阴影高度。
cardMaxElevation	CardView 的最大阴影高度。
cardPreventCornerOverlap	在 v20 及之前版本中为 CardView 添加内边距，以防止卡片内容与圆角重叠。 内容与圆角之间的交叉。
cardUseCompatPadding	在 API v21 及以上版本中也添加内边距，以保持与之前版本相同的测量值。可能是布尔值，如“true”或“false”。
contentPadding	Card 边缘与 CardView 子视图之间的内部填充。
contentPaddingBottom	卡片底边缘与 CardView 子元素之间的内边距。
contentPaddingLeft	Card 左边缘与 CardView 子视图之间的内部填充。
contentPaddingRight	CardView 的阴影高度。
cardElevation	Card 右边缘与 CardView 子视图之间的内部填充。
contentPaddingTop	Card 顶部边缘与 CardView 子视图之间的内部填充。

带有圆角背景和阴影的 FrameLayout。

CardView 在 Lollipop 版本上使用 elevation 属性实现阴影，在较旧平台上则回退到自定义的模拟阴影实现。

由于圆角裁剪开销较大，在 Lollipop 之前的平台上，CardView 不会裁剪与圆角相交的子视图。相反，它会添加填充以避免这种相交（参见 setPreventCornerOverlap(boolean) 以更改此行为）。

第14.1节：CardView 入门

CardView 是 Android 支持库的一个成员，提供卡片布局。

要将 CardView 添加到您的项目中，请在 build.gradle 的依赖项中添加以下行。

```
compile 'com.android.support:cardview-v7:25.1.1'
```

可以在 [here](#) 找到多个最新版本号

然后，您可以在布局中添加以下内容以获得一个卡片。

```
<android.support.v7.widget.CardView  
    xmlns:card_view="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
    <!-- 一个包含其他布局或视图的子布局 -->  
  
</android.support.v7.widget.CardView>
```

然后您可以在此内部添加其他布局，它们将被包含在卡片中。

此外，CardView 可以填充任何 UI 元素，并且可以通过代码进行操作。

```
<?xml version="1.0" encoding="utf-8"?>
```

Chapter 14: CardView

Parameter	Details
cardBackgroundColor	Background color for CardView.
cardCornerRadius	Corner radius for CardView.
cardElevation	Elevation for CardView.
cardMaxElevation	Maximum Elevation for CardView.
cardPreventCornerOverlap	Add padding to CardView on v20 and before to prevent intersections between the Card content and rounded corners.
cardUseCompatPadding	Add padding in API v21+ as well to have the same measurements with previous versions. May be a boolean value, such as "true" or "false".
contentPadding	Inner padding between the edges of the Card and children of the CardView.
contentPaddingBottom	Inner padding between the bottom edge of the Card and children of the CardView.
contentPaddingLeft	Inner padding between the left edge of the Card and children of the CardView.
contentPaddingRight	Elevation for CardView.
cardElevation	Inner padding between the right edge of the Card and children of the CardView.
contentPaddingTop	Inner padding between the top edge of the Card and children of the CardView.

A FrameLayout with a rounded corner background and shadow.

CardView uses elevation property on Lollipop for shadows and falls back to a custom emulated shadow implementation on older platforms.

Due to expensive nature of rounded corner clipping, on platforms before Lollipop, CardView does not clip its children that intersect with rounded corners. Instead, it adds padding to avoid such intersection (See setPreventCornerOverlap(boolean) to change this behavior).

Section 14.1: Getting Started with CardView

CardView is a member of the Android Support Library, and provides a layout for cards.

To add CardView to your project, add the following line to your `build.gradle` dependencies.

```
compile 'com.android.support:cardview-v7:25.1.1'
```

A number of the latest version may be found [here](#)

In your layout you can then add the following to get a card.

```
<android.support.v7.widget.CardView  
    xmlns:card_view="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
    <!-- one child layout containing other layouts or views -->  
  
</android.support.v7.widget.CardView>
```

You can then add other layouts inside this and they will be encompassed in a card.

Also, CardView can be populated with any UI element and manipulated from [code](#).

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_margin="5dp"
    card_view:cardBackgroundColor="#81C784"
    card_view:cardCornerRadius="12dp"
    card_view:cardElevation="3dp"
    card_view:contentPadding="4dp" >

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp" >

        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:id="@+id/item_image"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="16dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_title"
            android:layout_toRightOf="@+id/item_image"
            android:layout_alignParentTop="true"
            android:textSize="30sp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_detail"
            android:layout_toRightOf="@+id/item_image"
            android:layout_below="@+id/item_title"
            />

    </RelativeLayout>
</android.support.v7.widget.CardView>

```

第14.2节：添加涟漪动画

要在CardView中启用涟漪动画，请添加以下属性：

```

<android.support.v7.widget.CardView
    ...
    android:clickable="true"
    android:foreground="?android:attr/selectableItemBackground">
    ...
</android.support.v7.widget.CardView>

```

第14.3节：自定义CardView

CardView 提供了默认的阴影高度和圆角半径，使卡片在各处具有一致的外观

```

<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_margin="5dp"
    card_view:cardBackgroundColor="#81C784"
    card_view:cardCornerRadius="12dp"
    card_view:cardElevation="3dp"
    card_view:contentPadding="4dp" >

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp" >

        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:id="@+id/item_image"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="16dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_title"
            android:layout_toRightOf="@+id/item_image"
            android:layout_alignParentTop="true"
            android:textSize="30sp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_detail"
            android:layout_toRightOf="@+id/item_image"
            android:layout_below="@+id/item_title"
            />

    </RelativeLayout>
</android.support.v7.widget.CardView>

```

Section 14.2: Adding Ripple animation

To enable the ripple animation in a CardView, add the following attributes:

```

<android.support.v7.widget.CardView
    ...
    android:clickable="true"
    android:foreground="?android:attr/selectableItemBackground">
    ...
</android.support.v7.widget.CardView>

```

Section 14.3: Customizing the CardView

CardView provides a default elevation and corner radius so that cards have a consistent appearance across the

平台。

您可以使用xml文件中的这些属性自定义这些默认值：

1. card_view : cardElevation 属性用于在CardView中添加阴影高度。
2. card_view : cardBackgroundColor 属性用于自定义CardView的背景颜色（您可以设置任意颜色）。
3. card_view : cardCornerRadius 属性用于圆角CardView的四个边角
4. card_view : contentPadding 属性用于添加卡片与其子视图之间的内边距

注意：card_view是顶层父布局视图中定义的命名空间。

`xmlns:card_view="http://schemas.android.com/apk/res-auto"`

这里有一个示例：

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardElevation="4dp"
    card_view:cardBackgroundColor="@android:color/white"
    card_view:cardCornerRadius="8dp"
    card_view:contentPadding="16dp">

    <!-- 一个包含其他布局或视图的子布局 -->

</android.support.v7.widget.CardView>
```

你也可以通过编程方式实现：

```
card.setCardBackgroundColor(...);
card.setCardElevation(...);
card.setRadius(...);
card.setContentPadding();
```

查看官方 [javadoc](#)了解更多属性。

第14.4节：在CardView中使用图片作为背景（Lollipop之前设备的问题）

在CardView中使用图片/颜色作为背景时，如果默认卡片颜色为白色，边缘可能会出现轻微的白色内边距。这是由于CardView默认的圆角导致的。以下是在Lollipop之前设备中避免这些边距的方法。

我们需要在CardView中使用属性`card_view:cardPreventCornerOverlap="false"`。1). 在XML中使用以下代码片段。

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    card_view:cardPreventCornerOverlap="false"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/row_wallet_redeem_img"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:adjustViewBounds="true"
```

platforms.

You can customize these default values using these attributes in the xml file:

1. card_view:cardElevation attribute add elevation in CardView.
2. card_view:cardBackgroundColor attribute is used to customize background color of CardView's background(you can give any color).
3. card_view:cardCornerRadius attribute is used to curve 4 edges of CardView
4. card_view:contentPadding attribute add padding between card and children of card

Note: card_view is a namespace defined in topmost parent layout view.

`xmlns:card_view="http://schemas.android.com/apk/res-auto"`

Here an example:

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardElevation="4dp"
    card_view:cardBackgroundColor="@android:color/white"
    card_view:cardCornerRadius="8dp"
    card_view:contentPadding="16dp">

    <!-- one child layout containing other layouts or views -->

</android.support.v7.widget.CardView>
```

You can also do it programmatically using:

```
card.setCardBackgroundColor(...);
card.setCardElevation(...);
card.setRadius(...);
card.setContentPadding();
```

Check the [official javadoc](#) for additional properties.

Section 14.4: Using Images as Background in CardView (Pre-Lollipop device issues)

While using Image/Colour as an background in a CardView, You might end up with slight white paddings (If default Card colour is white) on the edges. This occurs due to the default rounded corners in the Card View. Here is how to avoid those margins in Pre-lollipop devices.

We need to use an attribute `card_view:cardPreventCornerOverlap="false"` in the CardView. 1). In XML use the following snippet.

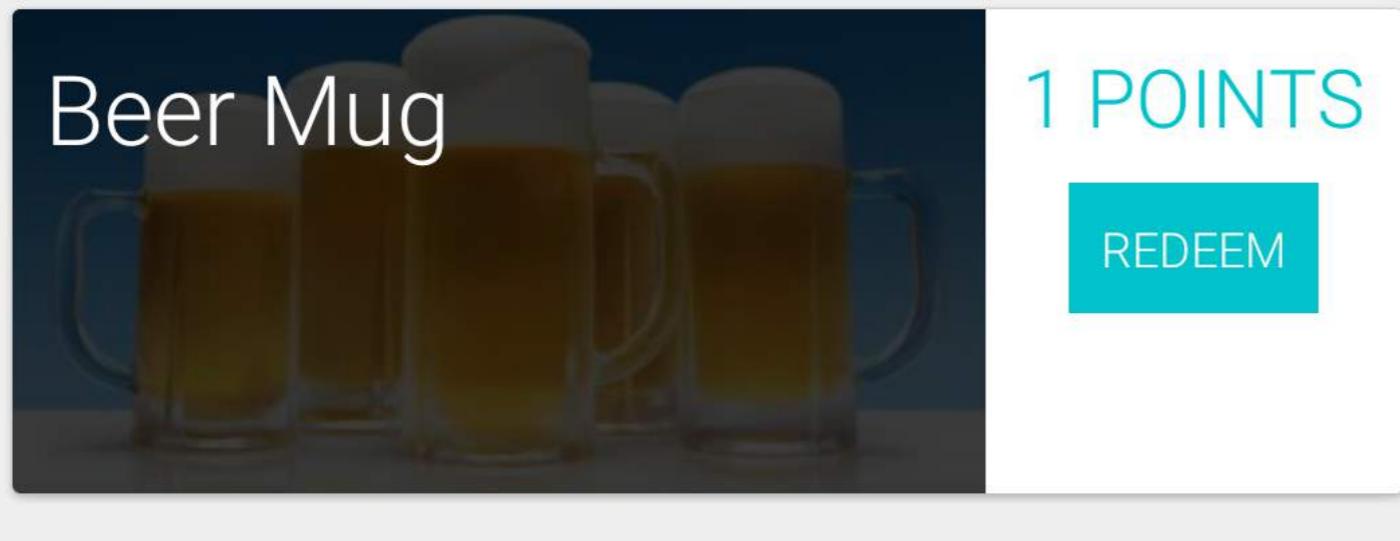
```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    card_view:cardPreventCornerOverlap="false"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/row_wallet_redeem_img"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:adjustViewBounds="true"
```

```
    android:scaleType="centerCrop"  
    android:src="@drawable/bg_image" />  
  
</android.support.v7.widget.CardView>
```

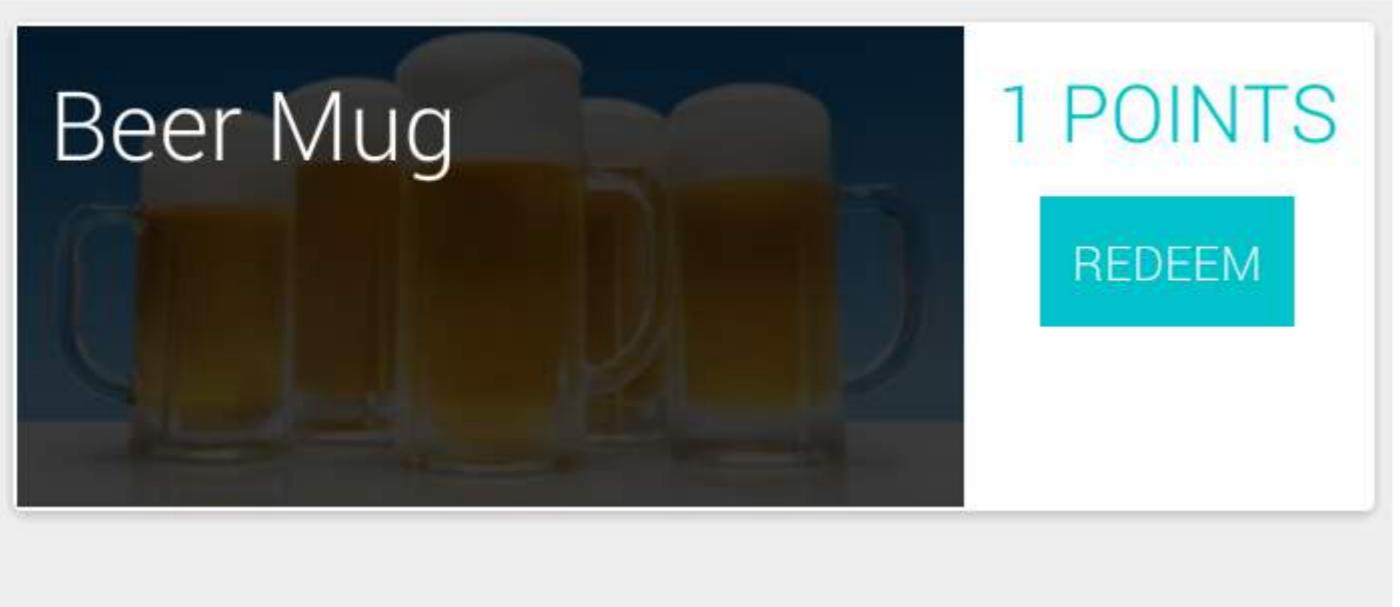
2. 在Java中这样写cardView.setPreventCornerOverlap(false).

这样做可以去除Card边缘不需要的内边距。以下是与此实现相关的一些视觉示例。

1 在API 21中带有图片背景的Card (完全正常)



2 在API 19中没有该属性的带图片背景的Card (注意图片周围的内边距)



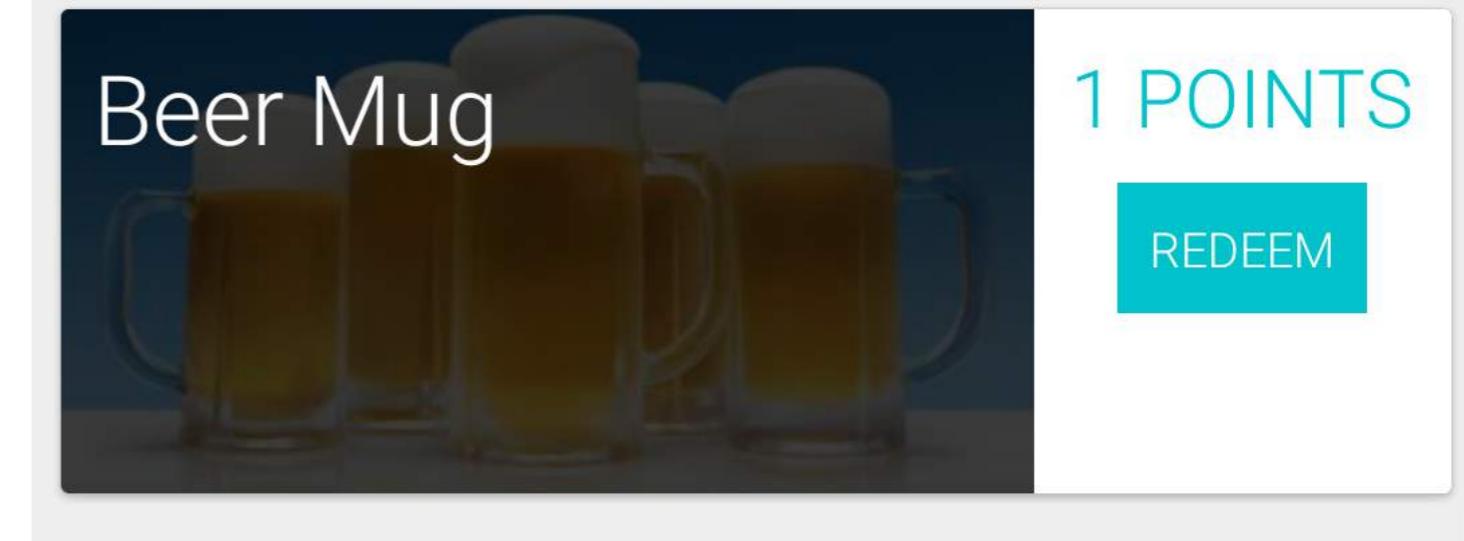
3 修复后的带图片背景的Card，在API 19中使用属性cardView.setPreventCornerOverlap(**false**) (问题已修复)

```
    android:scaleType="centerCrop"  
    android:src="@drawable/bg_image" />  
  
</android.support.v7.widget.CardView>
```

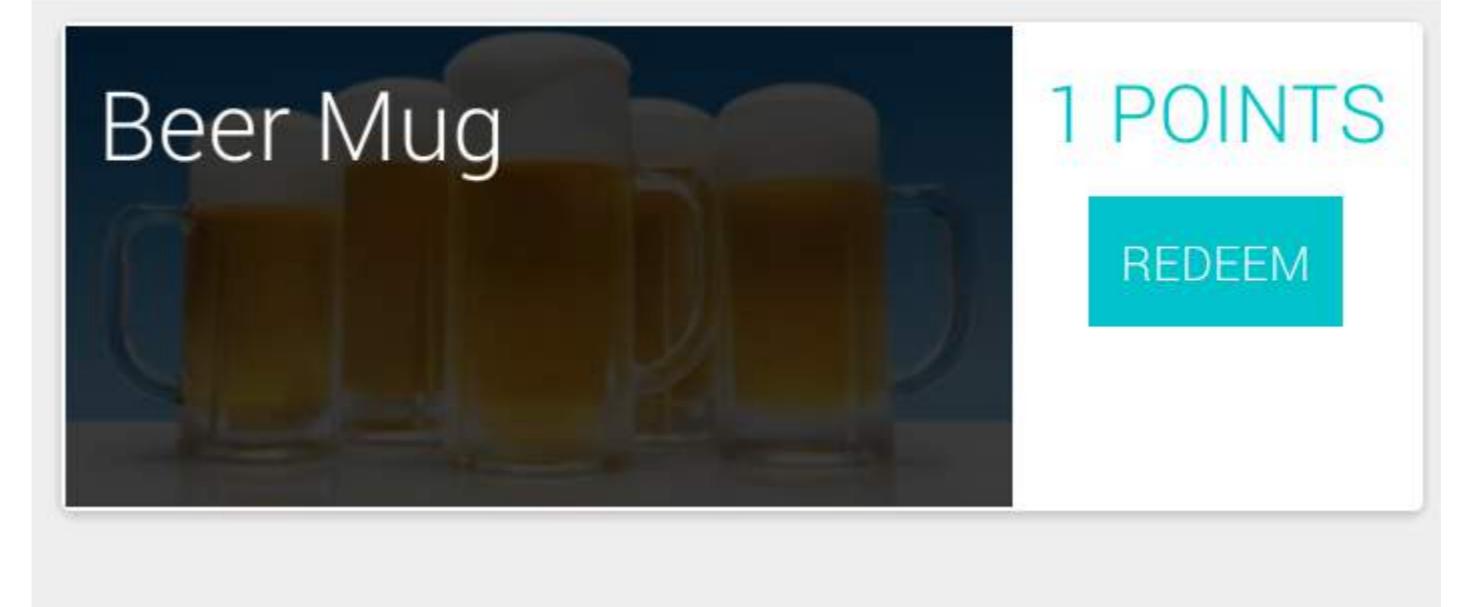
2. In Java like this cardView.setPreventCornerOverlap(**false**).

Doing so removes an unwanted padding on the Card's edges. Here are some visual examples related to this implementation.

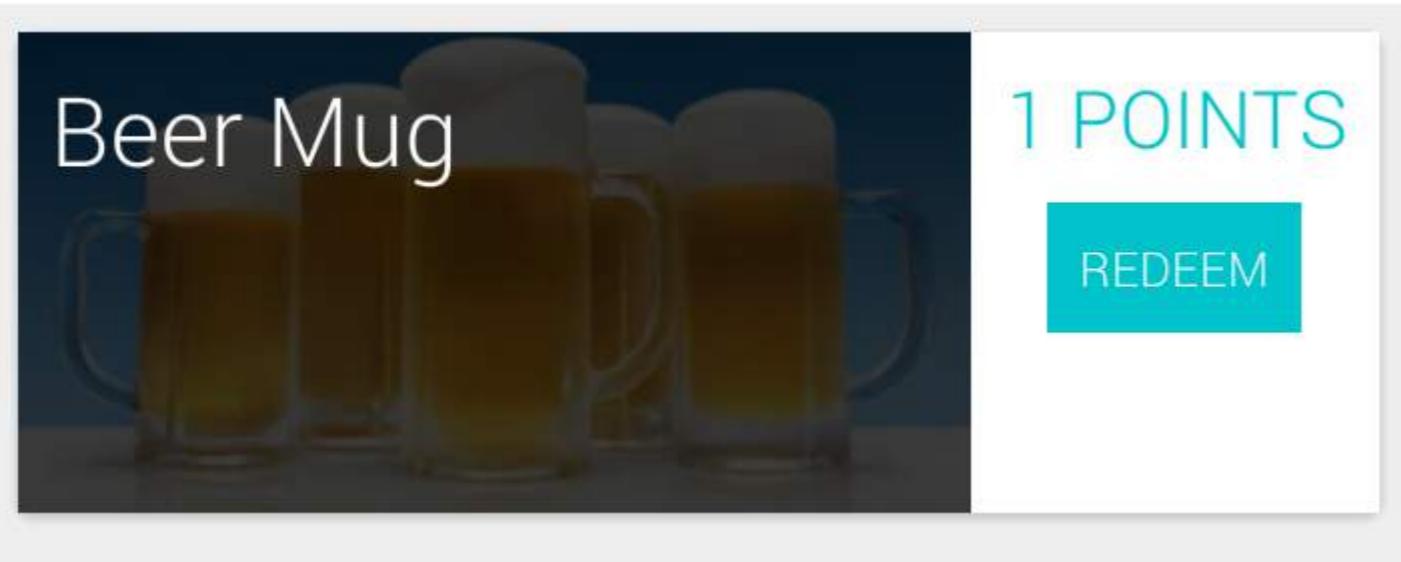
1 Card with image background in API 21 (perfectly fine)



2 Card with image background in API 19 without attribute (notice the paddings around image)



3 FIXED Card with image background in API 19 with attribute cardView.setPreventCornerOverlap(**false**) (Issue now fixed)

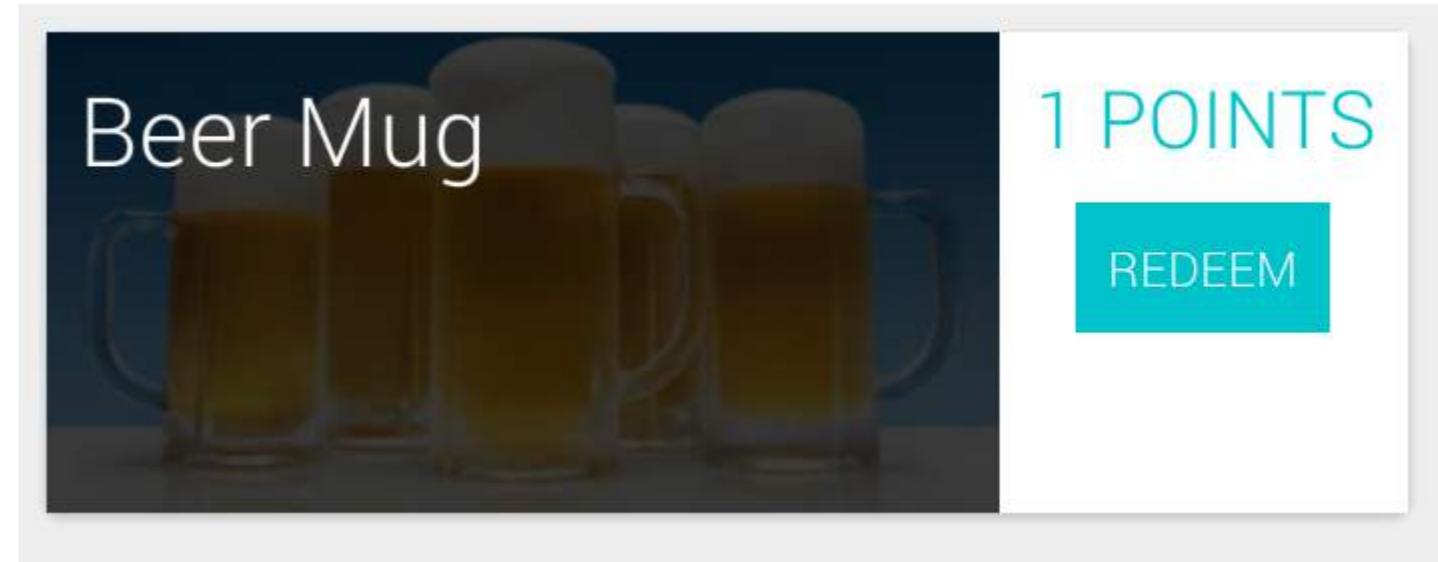


也可以阅读[文档这里](#)
原始SOF帖子[这里](#)

第14.5节：使用

TransitionDrawable动画CardView背景颜色

```
public void setCardColorTran(CardView card) {  
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};  
    TransitionDrawable trans = new TransitionDrawable(color);  
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {  
        card.setBackground(trans);  
    } 否则 {  
        card.setBackgroundDrawable(trans);  
    }  
    trans.startTransition(5000);  
}
```



Also read about this on [Documentation here](#)
Original SOF post [here](#)

Section 14.5: Animate CardView background color with TransitionDrawable

```
public void setCardColorTran(CardView card) {  
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};  
    TransitionDrawable trans = new TransitionDrawable(color);  
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {  
        card.setBackground(trans);  
    } else {  
        card.setBackgroundDrawable(trans);  
    }  
    trans.startTransition(5000);  
}
```

第15章：导航视图

第15.1节：如何添加导航视图

要使用导航视图，只需在build.gradle文件中添加依赖，如备注部分所述

然后在布局中添加NavigationView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

res/layout/nav_header_main.xml: 将显示在抽屉顶部的视图

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark"
    android:orientation="vertical"
    android:gravity="bottom">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:src="@android:drawable/sym_def_app_icon"
        android:id="@+id/imageView" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

Chapter 15: NavigationView

Section 15.1: How to add the NavigationView

To use a NavigationView just add the dependency in the build.gradle file as described in the remarks section

Then add the NavigationView in the layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

res/layout/nav_header_main.xml: The view which will be displayed on the top of the drawer

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark"
    android:orientation="vertical"
    android:gravity="bottom">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:src="@android:drawable/sym_def_app_icon"
        android:id="@+id/imageView" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
    android:paddingTop="@dimen/nav_header_vertical_spacing"
    android:text="Android Studio"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="android.studio@android.com"
    android:id="@+id/textView" />
```

res/layout/app_bar_main.xml 工具栏的抽象层，用于将其与内容分离：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="eu.rekisoft.playground.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

res/layout/content_main.xml 活动的真实内容仅用于演示，这里你可以放置你的常规布局xml：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingTop="@dimen/nav_header_vertical_spacing"
    android:text="Android Studio"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="android.studio@android.com"
    android:id="@+id/textView" />
```

res/layout/app_bar_main.xml An abstraction layer for the toolbar to separate it from the content:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="eu.rekisoft.playground.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

res/layout/content_main.xml The real content of the activity just for demo, here you would put your normal layout xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```

        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:paddingBottom="@dimen/activity_vertical_margin"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:showIn="@layout/app_bar_main"
        tools:context="eu.rekisoft.playground.MainActivity">

    <TextView
        android:text="Hello World!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>

```

将您的菜单文件定义为res/menu/activity_main_drawer.xml :

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="导入" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="图库" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="幻灯片" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="工具" />
    </group>

    <item android:title="通讯">
        <menu >
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="分享" />
            <item
                android:id="@+id/nav_send"
                android:icon="@drawable/ic_menu_send"
                android:title="发送" />
        </menu>
    </item>
</menu>

```

最后是java/main/eu/rekisoft/playground/MainActivity.java:

```

public class MainActivity extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelected {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

```

        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:paddingBottom="@dimen/activity_vertical_margin"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:showIn="@layout/app_bar_main"
        tools:context="eu.rekisoft.playground.MainActivity">

    <TextView
        android:text="Hello World!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>

```

Define your menu file as res/menu/activity_main_drawer.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="Gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="Slideshow" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="Tools" />
    </group>

    <item android:title="Communicate">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="Share" />
            <item
                android:id="@+id/nav_send"
                android:icon="@drawable/ic_menu_send"
                android:title="Send" />
        </menu>
    </item>
</menu>

```

And finally the java/main/eu/rekisoft/playground/MainActivity.java:

```

public class MainActivity extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelected {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

```

Toolbar 工具栏 = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

FloatingActionButton 悬浮按钮 = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "用你自己的操作替换", Snackbar.LENGTH_LONG)
            .setAction("操作", null).show();
    }
});

DrawerLayout 抽屉布局 = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle 切换按钮 = new ActionBarDrawerToggle(
    this, 抽屉布局, 工具栏, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
抽屉布局.setDrawerListener(切换按钮);
切换按钮.syncState();

NavigationView 导航视图 = (NavigationView) findViewById(R.id.nav_view);
导航视图.setNavigationItemSelectedListener(this);
}

@Override
public void onBackPressed() {
    DrawerLayout 抽屉布局 = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (抽屉布局.isDrawerOpen(GravityCompat.START)) {
        抽屉布局.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu 菜单) {
    // 加载菜单；如果存在，则将项目添加到操作栏。
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // 处理操作栏的项目点击事件。只要你在 AndroidManifest.xml 中指定了
    // 父活动，操作栏会自动处理主页/向上按钮的点击事件。
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // 处理导航视图的项目点击事件。
    switch(item.getItemId()) {/*...*/}
}

DrawerLayout 抽屉布局 = (DrawerLayout) findViewById(R.id.drawer_layout);

```

```

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
    }
});

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
    this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
drawer.setDrawerListener(toggle);
toggle.syncState();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

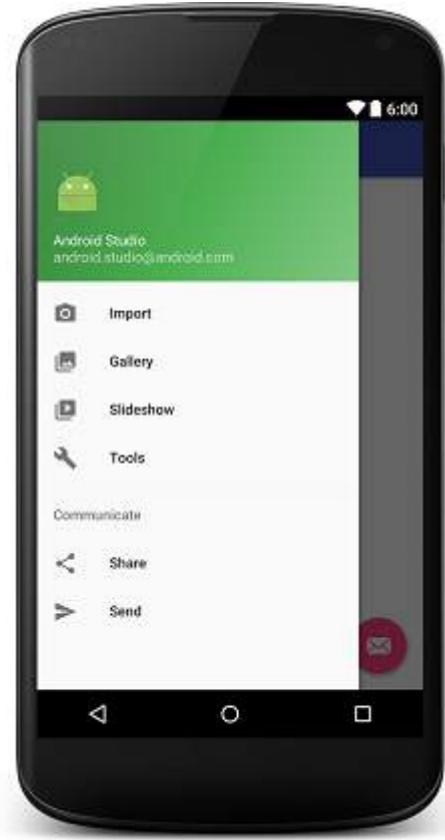
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    switch(item.getItemId()) {/*...*/}
}

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);

```

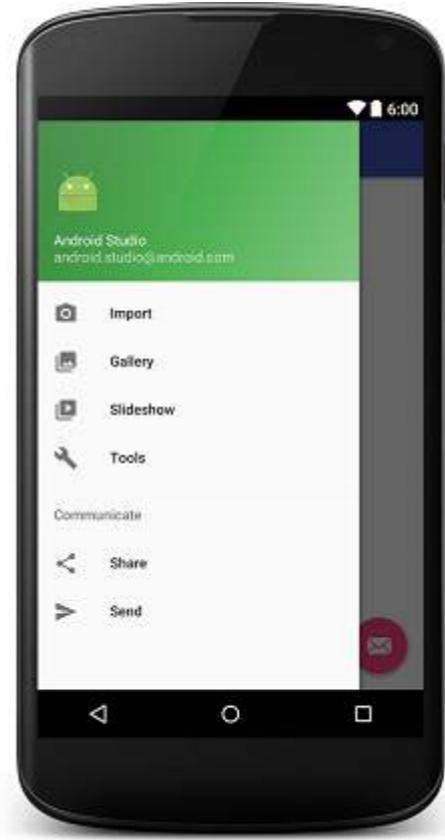
```
drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

它将如下所示：



```
drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

It will look like this:



第15.2节：在菜单元素中添加下划线

每个组以一条分隔线结束。如果菜单中的每个项目都有自己的组，你将实现所需的图形输出。只有当不同组具有不同的 android:id 时，这才有效。此外，在 menu.xml 中，记得为单个项目设置 android:checkable="true"，为一组项目设置 android:checkableBehavior="single"。

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/pos_item_help"
        android:checkable="true"
        android:title="帮助" />
    <item
        android:id="@+id/pos_item_pos"
        android:checkable="true"
        android:title="收银台" />

    <item
        android:id="@+id/pos_item_orders"
        android:checkable="true"
        android:title="订单" />

    <group
        android:id="@+id/group"
        android:checkableBehavior="single">
```

Section 15.2: Add underline in menu elements

Each group ends with a line separator. If each item in your menu has its own group you will achieve the desired graphical output. It will work only if your different groups have different android:id. Also, in menu.xml remember to mention android:checkable="true" for single item and android:checkableBehavior="single" for a group of items.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/pos_item_help"
        android:checkable="true"
        android:title="Help" />
    <item
        android:id="@+id/pos_item_pos"
        android:checkable="true"
        android:title="POS" />

    <item
        android:id="@+id/pos_item_orders"
        android:checkable="true"
        android:title="Orders" />

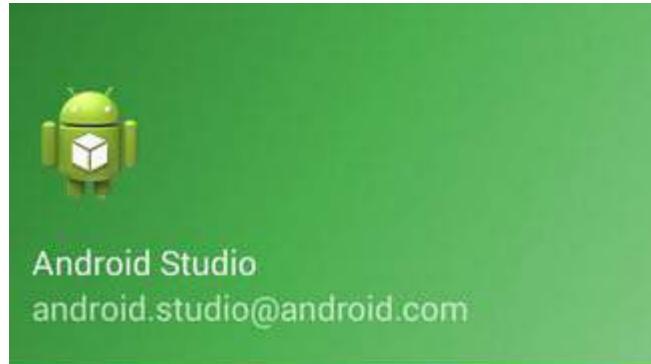
    <group
        android:id="@+id/group"
        android:checkableBehavior="single">
```

```

<item
    android:id="@+id/menu_nav_home"
    android:icon="@drawable/ic_home_black_24dp"
    android:title="@string/menu_nav_home" />
</group>

.....
</menu>

```



Import

Gallery

Slideshow

Tools

Share

Send

第15.3节：向菜单添加分隔符

访问NavigationView中的RecyclerView并向其添加ItemDecoration。

```

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));

```

DividerItemDecoration 的代码

```

<item
    android:id="@+id/menu_nav_home"
    android:icon="@drawable/ic_home_black_24dp"
    android:title="@string/menu_nav_home" />
</group>

.....
</menu>

```



Import

Gallery

Slideshow

Tools

Share

Send

Section 15.3: Add separators to menu

Access the RecyclerView in the NavigationView and add ItemDecoration to it.

```

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));

```

Code for DividerItemDecoration

```

public class DividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{android.R.attr.listDivider};

    private Drawable mDivider;

    public DividerItemDecoration(Context context) {
        final TypedArray styledAttributes = context.obtainStyledAttributes(ATTRS);
        mDivider = styledAttributes.getDrawable(0);
        styledAttributes.recycle();
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        int left = parent.getPaddingLeft();
        int right = parent.getWidth() - parent.getPaddingRight();

        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

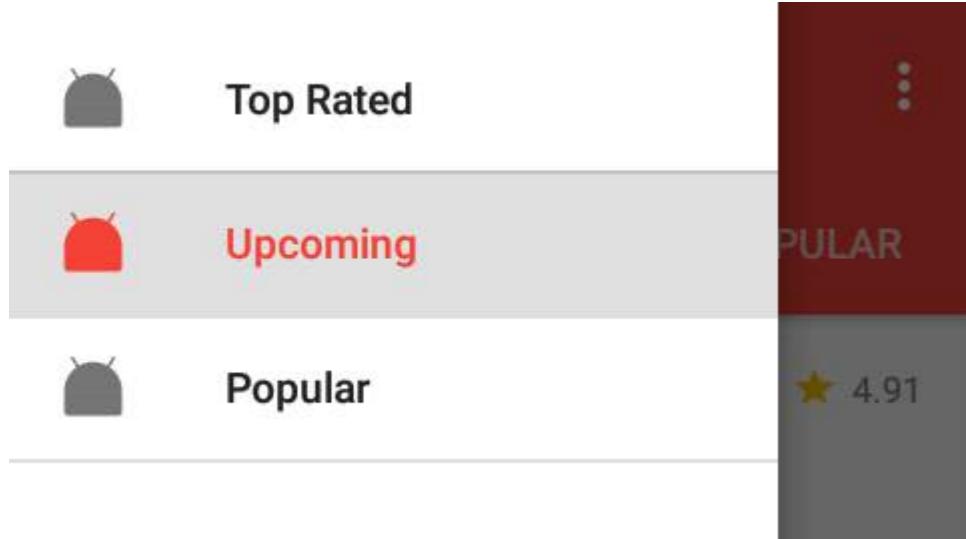
            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();

            int top = child.getBottom() + params.bottomMargin;
            int bottom = top + mDivider.getIntrinsicHeight();

            mDivider.setBounds(left, top, right, bottom);
            mDivider.draw(c);
        }
    }
}

```

预览：



第15.4节：使用默认DividerItemDecoration 添加菜单分割线

只需使用默认的DividerItemDecoration类：

```

NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);

```

```

public class DividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{android.R.attr.listDivider};

    private Drawable mDivider;

    public DividerItemDecoration(Context context) {
        final TypedArray styledAttributes = context.obtainStyledAttributes(ATTRS);
        mDivider = styledAttributes.getDrawable(0);
        styledAttributes.recycle();
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        int left = parent.getPaddingLeft();
        int right = parent.getWidth() - parent.getPaddingRight();

        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

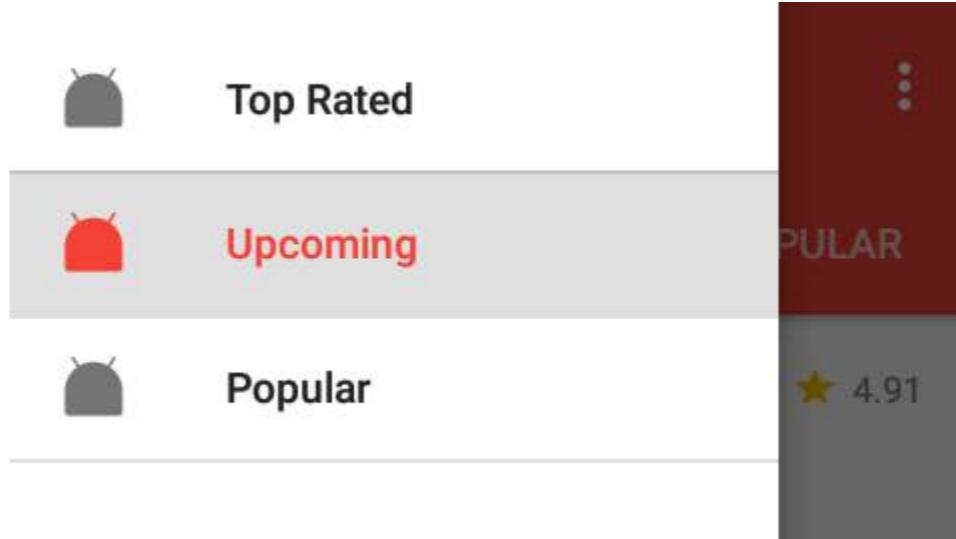
            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();

            int top = child.getBottom() + params.bottomMargin;
            int bottom = top + mDivider.getIntrinsicHeight();

            mDivider.setBounds(left, top, right, bottom);
            mDivider.draw(c);
        }
    }
}

```

Preview:



Section 15.4: Add menu Divider using default DividerItemDecoration

Just use default DividerItemDecoration class :

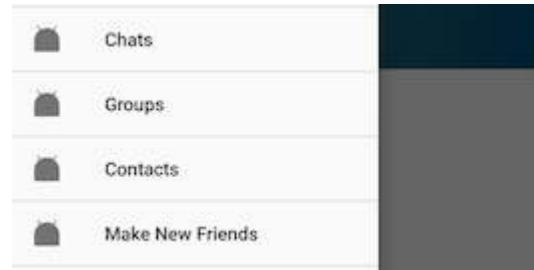
```

NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);

```

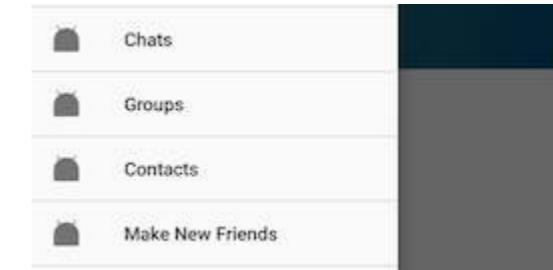
```
navMenuView.addItemDecoration(new DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

预览 :



```
navMenuView.addItemDecoration(new DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Preview :



第16章：RecyclerView

参数

适配器 RecyclerView.Adapter的子类，负责提供表示数据集中的项目的视图

位置 适配器中数据项的位置

索引 调用getChildAt(int)时使用的附加子视图的索引。与位置(Position)相对

绑定 准备子视图以显示与适配器中某个位置对应的数据的过程

之前用于显示特定适配器位置数据的视图可能会被放入缓存中

回收 (视图) 以便稍后重用，重新显示相同类型的数据。这可以通过跳过初始布局膨胀或构建，显著提升性能

废弃 (视图) 在布局过程中进入临时分离状态的子视图。废弃视图可以在不完全脱离父RecyclerView的情况下被重用，如果不需要重新绑定则保持不变，如果视图被认为是脏的，则由适配器修改

脏 (视图) 必须由适配器重新绑定后才能显示的子视图

详情

RecyclerView是List View的更高级版本，具有更好的性能和更多功能。

第16.1节：添加RecyclerView

按照备注部分的说明添加依赖项，然后在布局中添加一个RecyclerView：

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/my_recycler_view"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

一旦你在布局中添加了一个RecyclerView控件，获取该对象的句柄，连接一个布局管理器，并附加一个用于显示数据的适配器：

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);  
  
// 设置布局管理器 (此示例中为LinearLayoutManager)  
  
mLayoutManager = new LinearLayoutManager(getApplicationContext());  
mRecyclerView.setLayoutManager(mLayoutManager);  
  
// 指定适配器  
mAdapter = new MyAdapter(myDataset);  
mRecyclerView.setAdapter(mAdapter);
```

或者通过在xml中添加以下行来简单设置布局管理器：

```
xmlns:app="http://schemas.android.com/apk/res-auto"  
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

如果你知道RecyclerView的内容变化不会改变RecyclerView的布局大小，使用以下代码可以提升组件的性能。如果RecyclerView具有固定大小，它就知道RecyclerView本身不会因为其子项而调整大小，因此它根本不会调用请求布局。它只是自行处理变化。如果使父视图无效，无论是协调器、布局还是其他什么。（你甚至可以在设置LayoutManager和Adapter之前使用此方法）：

```
mRecyclerView.setHasFixedSize(true);
```

Chapter 16: RecyclerView

Parameter

Adapter A subclass of RecyclerView.Adapter responsible for providing views that represent items in a data set

Position The position of a data item within an Adapter

Index The index of an attached child view as used in a call to getChildAt(int). Contrast with Position

Binding The process of preparing a child view to display data corresponding to a position within the adapter
A view previously used to display data for a specific adapter position may be placed in a cache for Recycle (view) later reuse to display the same type of data again later. This can drastically improve performance by skipping initial layout inflation or construction

Scrap (view) A child view that has entered into a temporarily detached state during layout. Scrap views may be reused without becoming fully detached from the parent RecyclerView, either unmodified if no rebinding is required or modified by the adapter if the view was considered dirty

Dirty (view) A child view that must be rebound by the adapter before being displayed

RecyclerView is a more advanced version of List View with improved performance and additional features.

Section 16.1: Adding a RecyclerView

Add the dependency as described in the Remark section, then add a RecyclerView to your layout:

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/my_recycler_view"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

Once you have added a RecyclerView widget to your layout, obtain a handle to the object, connect it to a layout manager and attach an adapter for the data to be displayed:

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);  
  
// set a layout manager (LinearLayoutManager in this example)  
  
mLayoutManager = new LinearLayoutManager(getApplicationContext());  
mRecyclerView.setLayoutManager(mLayoutManager);  
  
// specify an adapter  
mAdapter = new MyAdapter(myDataset);  
mRecyclerView.setAdapter(mAdapter);
```

Or simply setup layout manager from xml by adding this lines:

```
xmlns:app="http://schemas.android.com/apk/res-auto"  
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

If you know that changes in content of the RecyclerView won't change the layout size of the RecyclerView, use the following code to improve the performance of the component. If RecyclerView has a fixed size, it knows that RecyclerView itself will not resize due to its children, so it doesn't call request layout at all. It just handles the change itself. If invalidating whatever the parent is, the coordinator, layout, or whatever. (you can use this method even before setting LayoutManager and Adapter):

```
mRecyclerView.setHasFixedSize(true);
```

RecyclerView 提供了这些内置的布局管理器供使用。因此你可以使用RecyclerView创建列表、网格和错列网格：

1. [LinearLayoutManager](#) 显示垂直或水平滚动的列表项。
2. [GridLayoutManager](#) 显示网格中的项。
3. [StaggeredGridLayoutManager](#) 显示错列网格中的项。

第16.2节：更流畅的项目加载

如果你的RecyclerView中的项目从网络加载数据（通常是图片）或执行其他处理，这可能会花费相当长的时间，导致屏幕上的项目未完全加载。为避免这种情况，你可以扩展现有的LinearLayoutManager，在项目出现在屏幕上之前预加载一定数量的项目：

```
package com.example;

import android.content.Context;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.OrientationHelper;
import android.support.v7.widget.RecyclerView;

/**
 * 一个预加载屏幕外项目的线性布局管理器。
 * <p>
 * 预加载在某些情况下非常有用，比如项目可能需要一些时间才能完全加载，通常是因为它们包含地图
 * 、图片或其他需要通过网络请求完成后才能显示的内容。
 *
 * <p>
 * 默认情况下，该布局会加载额外一页的项目，
 * 一页是一个像素度量，相当于屏幕上RecyclerView的大小。
 * 可以通过相关构造函数或{@link #setPages(int)}方法来更改此设置。
 *
 */
public class PreLoadingLinearLayoutManager extends LinearLayoutManager {
    private int mPages = 1;
    private OrientationHelper mOrientationHelper;

    public PreLoadingLinearLayoutManager(Context context) {
        super(context);
    }

    public PreLoadingLinearLayoutManager(Context context, int pages) {
        super(context);
        this.mPages = pages;
    }

    public PreLoadingLinearLayoutManager(Context context, int orientation, boolean reverseLayout) {
        super(context, orientation, reverseLayout);
    }

    @Override
    public void setOrientation(int orientation) {
        super.setOrientation(orientation);
        mOrientationHelper = null;
    }

    /**
     * 设置将要预加载的布局页数,

```

RecyclerView provides these built-in layout managers to use. So you can create a list, a grid and a staggered grid using RecyclerView:

1. [LinearLayoutManager](#) shows items in a vertical or horizontal scrolling list.
2. [GridLayoutManager](#) shows items in a grid.
3. [StaggeredGridLayoutManager](#) shows items in a staggered grid.

Section 16.2: Smoother loading of items

If the items in your RecyclerView load data from the network (commonly images) or carry out other processing, that can take a significant amount of time and you may end up with items on-screen but not fully loaded. To avoid this you can extend the existing LinearLayoutManager to preload a number of items before they become visible on-screen:

```
package com.example;

import android.content.Context;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.OrientationHelper;
import android.support.v7.widget.RecyclerView;

/**
 * A LinearLayoutManager that preloads items off-screen.
 * <p>
 * Preloading is useful in situations where items might take some time to load
 * fully, commonly because they have maps, images or other items that require
 * network requests to complete before they can be displayed.
 *
 * <p>
 * By default, this layout will load a single additional page's worth of items,
 * a page being a pixel measure equivalent to the on-screen size of the
 * recycler view. This can be altered using the relevant constructor, or
 * through the {@link #setPages(int)} method.
 *
 */
public class PreLoadingLinearLayoutManager extends LinearLayoutManager {
    private int mPages = 1;
    private OrientationHelper mOrientationHelper;

    public PreLoadingLinearLayoutManager(Context context) {
        super(context);
    }

    public PreLoadingLinearLayoutManager(Context context, int pages) {
        super(context);
        this.mPages = pages;
    }

    public PreLoadingLinearLayoutManager(Context context, int orientation, boolean reverseLayout) {
        super(context, orientation, reverseLayout);
    }

    @Override
    public void setOrientation(int orientation) {
        super.setOrientation(orientation);
        mOrientationHelper = null;
    }

    /**
     * Set the number of pages of layout that will be preloaded off-screen,
```

```

* 一页是与屏幕上RecyclerView大小等效的像素度量。
* @param pages 页数；可以为{@code 0}以禁用预加载
 */
public void setPages(final int pages) {
    this.mPages = pages;
}

@Override
protected int getExtraLayoutSpace(final RecyclerView.State state) {
    if (mOrientationHelper == null) {
        mOrientationHelper = OrientationHelper.createOrientationHelper(this, getOrientation());
    }
    return mOrientationHelper.getTotalSpace() * mPages;
}

```

第16.3节：带DataBinding的RecyclerView

这是一个通用的ViewHolder类，可以与任何DataBinding布局一起使用。这里创建了一个特定的 [ViewDataBinding](#) 类的实例，使用了被膨胀的View对象和 [DataBindingUtil](#) 工具类。

```

import android.databinding.DataBindingUtil;
import android.support.v7.widget.RecyclerView;
import android.view.View;

public class BindingViewHolder<T> extends RecyclerView.ViewHolder{

    private final T binding;

    public BindingViewHolder(View itemView) {
        super(itemView);
        binding = (T)DataBindingUtil.bind(itemView);
    }

    public T getBinding() {
        return binding;
    }
}

```

创建此类后，您可以在布局文件中使用<layout>标签来启用该布局的数据绑定，示例如下：

文件名: my_item.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="item"
            type="ItemModel" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"

```

```

* a page being a pixel measure equivalent to the on-screen size of the
* recycler view.
* @param pages the number of pages; can be {@code 0} to disable preloading
*/
public void setPages(final int pages) {
    this.mPages = pages;
}

@Override
protected int getExtraLayoutSpace(final RecyclerView.State state) {
    if (mOrientationHelper == null) {
        mOrientationHelper = OrientationHelper.createOrientationHelper(this, getOrientation());
    }
    return mOrientationHelper.getTotalSpace() * mPages;
}

```

Section 16.3: RecyclerView with DataBinding

Here is a generic ViewHolder class that you can use with any DataBinding layout. Here an instance of particular [ViewDataBinding](#) class is created using the inflated [View](#) object and [DataBindingUtil](#) utility class.

```

import android.databinding.DataBindingUtil;
import android.support.v7.widget.RecyclerView;
import android.view.View;

public class BindingViewHolder<T> extends RecyclerView.ViewHolder{

    private final T binding;

    public BindingViewHolder(View itemView) {
        super(itemView);
        binding = (T)DataBindingUtil.bind(itemView);
    }

    public T getBinding() {
        return binding;
    }
}

```

After creating this class you can use the <layout> in your layout file to enable databinding for that layout like this:

file name: my_item.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="item"
            type="ItemModel" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"

```

```

        android:text="@{item.itemLabel}" />
    </LinearLayout>
</layout>

```

这是你的示例数据模型：

```

public class ItemModel {
    public String itemLabel;
}

```

默认情况下，Android 数据绑定库会根据布局文件名生成一个ViewDataBinding类，将文件名转换为帕斯卡命名法并加上“Binding”后缀。对于本例，布局文件my_item.xml对应的类名为MyItemBinding。该 Binding 类还会包含一个设置器方法，用于设置布局文件中定义为数据的对象（本例中为ItemModel）。

现在我们已经准备好所有部分，可以这样实现我们的适配器：

```

class MyAdapter extends RecyclerView.Adapter<BindingViewHolder<MyItemBinding>>{
    ArrayList<ItemModel> items = new ArrayList<>();

    public MyAdapter(ArrayList<ItemModel> items) {
        this.items = items;
    }

    @Override public BindingViewHolder<MyItemBinding> onCreateViewHolder(ViewGroup parent, int viewType) {
        return new
        BindingViewHolder<>(LayoutInflater.from(parent.getContext()).inflate(R.layout.my_item, parent,
false));
    }

    @Override public void onBindViewHolder(BindingViewHolder<ItemModel> holder, int position) {
        holder.getBinding().setItemModel(items.get(position));
        holder.getBinding().executePendingBindings();
    }

    @Override public int getItemCount() {
        return items.size();
    }
}

```

第16.4节：动画数据变更

如果使用了任何“notify”方法，除了
notifyDataSetChanged之外，RecyclerView将执行相应的动画；这包括notifyItemChanged、notifyItemInserted、notifyItemMoved、
notifyItemRemoved等。

适配器应继承此类，而不是RecyclerView.Adapter。

```

import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder> extends Recycl
erView.Adapter<VH> {
    protected List<T> models;
}

```

```

        android:text="@{item.itemLabel}" />
    </LinearLayout>
</layout>

```

and here is your sample dataModel:

```

public class ItemModel {
    public String itemLabel;
}

```

By default, Android Data Binding library generates a ViewDataBinding class based on the layout file name, converting it to Pascal case and suffixing "Binding" to it. For this example it would be MyItemBinding for the layout file my_item.xml. That Binding class would also have a setter method to set the object defined as data in the layout file (ItemModel for this example).

Now that we have all the pieces we can implement our adapter like this:

```

class MyAdapter extends RecyclerView.Adapter<BindingViewHolder<MyItemBinding>>{
    ArrayList<ItemModel> items = new ArrayList<>();

    public MyAdapter(ArrayList<ItemModel> items) {
        this.items = items;
    }

    @Override public BindingViewHolder<MyItemBinding> onCreateViewHolder(ViewGroup parent, int viewType) {
        return new
        BindingViewHolder<>(LayoutInflater.from(parent.getContext()).inflate(R.layout.my_item, parent,
false));
    }

    @Override public void onBindViewHolder(BindingViewHolder<ItemModel> holder, int position) {
        holder.getBinding().setItemModel(items.get(position));
        holder.getBinding().executePendingBindings();
    }

    @Override public int getItemCount() {
        return items.size();
    }
}

```

Section 16.4: Animate data change

RecyclerView will perform a relevant animation if any of the "notify" methods are used except for
notifyDataSetChanged; this includes notifyItemChanged, notifyItemInserted, notifyItemMoved,
notifyItemRemoved, etc.

The adapter should extend this class instead of RecyclerView.Adapter.

```

import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder>
    extends RecyclerView.Adapter<VH> {
    protected List<T> models;
}

```

```

protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
    this.models = models;
}

//设置新的模型。
public void setModels(@NonNull final List<T> models) {
    applyAndAnimateRemovals(models);
    applyAndAnimateAdditions(models);
    applyAndAnimateMovedItems(models);
}

//在指定位置移除一个项目并通知更改。
private T 移除项目(int 位置) {
    final T 模型 = 模型列表.移除(位置);
    通知项目已移除(位置);
    return 模型;
}

//在指定位置添加项目并通知更改。
private void 添加项目(int 位置, T 模型) {
    模型列表.添加(位置, 模型);
    通知项目已插入(位置);
}

//将项目从 fromPosition 移动到 toPosition 并通知更改。
private void 移动项目(int 起始位置, int 目标位置) {
    final T 模型 = 模型列表.移除(起始位置);
    模型列表.添加(目标位置, 模型);
    通知项目已移动(起始位置, 目标位置);
}

//移除新模型中不再存在的项目。
private void 应用并动画移除(@NonNull final List<T> 新模型列表) {
    for (int i = 模型列表.大小() - 1; i >= 0; i--) {
        final T 模型 = 模型列表.获取(i);
        if (!新模型列表.包含(模型)) {
            移除项目(i);
        }
    }
}

//添加旧模型中不存在的项目。
private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
    for (int i = 0, count = newTs.size(); i < count; i++) {
        final T model = newTs.get(i);
        if (!models.contains(model)) {
            addItem(i, model);
        }
    }
}

//移动位置发生变化的项目。
private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
    for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
        final T model = newTs.get(toPosition);
        final int fromPosition = models.indexOf(model);
        if (fromPosition >= 0 && fromPosition != toPosition) {
            moveItem(fromPosition, toPosition);
        }
    }
}

```

```

protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
    this.models = models;
}

//Set new models.
public void setModels(@NonNull final List<T> models) {
    applyAndAnimateRemovals(models);
    applyAndAnimateAdditions(models);
    applyAndAnimateMovedItems(models);
}

//Remove an item at position and notify changes.
private T removeItem(int position) {
    final T model = models.remove(position);
    notifyItemRemoved(position);
    return model;
}

//Add an item at position and notify changes.
private void addItem(int position, T model) {
    models.add(position, model);
    notifyItemInserted(position);
}

//Move an item at fromPosition to toPosition and notify changes.
private void moveItem(int fromPosition, int toPosition) {
    final T model = models.remove(fromPosition);
    models.add(toPosition, model);
    notifyItemMoved(fromPosition, toPosition);
}

//Remove items that no longer exist in the new models.
private void applyAndAnimateRemovals(@NonNull final List<T> newTs) {
    for (int i = models.size() - 1; i >= 0; i--) {
        final T model = models.get(i);
        if (!newTs.contains(model)) {
            removeItem(i);
        }
    }
}

//Add items that do not exist in the old models.
private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
    for (int i = 0, count = newTs.size(); i < count; i++) {
        final T model = newTs.get(i);
        if (!models.contains(model)) {
            addItem(i, model);
        }
    }
}

//Move items that have changed their position.
private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
    for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
        final T model = newTs.get(toPosition);
        final int fromPosition = models.indexOf(model);
        if (fromPosition >= 0 && fromPosition != toPosition) {
            moveItem(fromPosition, toPosition);
        }
    }
}

```

你不应该在适配器中对 setModels 和 List 使用相同的 List。

你将 models 声明为全局变量。DataModel 只是一个示例类。

```
private List<DataModel> models;  
private YourAdapter adapter;
```

在传递给适配器之前初始化 models。YourAdapter 是 AnimatedRecyclerAdapter 的实现类。

```
models = new ArrayList<>();  
//添加模型  
models.add(new DataModel());  
//不要直接传递models。否则，当你修改全局models时，  
//你也会修改适配器中的models。  
//adapter = new YourAdapter(models); <- 这是错误的。  
adapter = new YourAdapter(new ArrayList(models));
```

在更新全局models后调用此方法。

```
adapter.setModels(new ArrayList(models));
```

如果你没有重写equals，所有比较都是通过引用进行的。

使用SortedList的示例

Android在引入RecyclerView后不久引入了SortedList类。该类处理所有对RecyclerView.Adapter的“notify”方法调用，以确保动画的正确性，甚至允许批量处理多个更改，从而避免动画抖动。

```
import android.support.v7.util.SortedList;  
import android.support.v7.widget.RecyclerView;  
import android.support.v7.widget.util.SortedListAdapterCallback;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import java.util.List;  
  
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {  
  
    private SortedList<DataModel> mSortedList;  
  
    class ViewHolder extends RecyclerView.ViewHolder {  
  
        TextView text;  
        CheckBox checkBox;  
  
        ViewHolder(View itemView){  
            super(itemView);  
  
            //在这里初始化你的代码...  
        }  
  
        void setDataModel(DataModel model) {  
            //使用传入的数据模型更新你的UI...  
        }  
    }  
}
```

You should **NOT** use the same [List](#) for setModels and [List](#) in the adapter.

You declare models as global variables. DataModel is a dummy class only.

```
private List<DataModel> models;  
private YourAdapter adapter;
```

Initialize models before pass it to adapter. YourAdapter is the implementation of AnimatedRecyclerAdapter.

```
models = new ArrayList<>();  
//Add models  
models.add(new DataModel());  
//Do NOT pass the models directly. Otherwise, when you modify global models,  
//you will also modify models in adapter.  
//adapter = new YourAdapter(models); <- This is wrong.  
adapter = new YourAdapter(new ArrayList(models));
```

Call this after you have updated your global models.

```
adapter.setModels(new ArrayList(models));
```

If you do not override equals, all the comparison is compared by reference.

Example using SortedList

Android introduced the [SortedList](#) class soon after RecyclerView was introduced. This class handles all 'notify' method calls to the [RecyclerView.Adapter](#) to ensure proper animation, and even allows batching multiple changes, so the animations don't jitter.

```
import android.support.v7.util.SortedList;  
import android.support.v7.widget.RecyclerView;  
import android.support.v7.widget.util.SortedListAdapterCallback;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import java.util.List;  
  
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {  
  
    private SortedList<DataModel> mSortedList;  
  
    class ViewHolder extends RecyclerView.ViewHolder {  
  
        TextView text;  
        CheckBox checkBox;  
  
        ViewHolder(View itemView){  
            super(itemView);  
  
            //Initiate your code here...  
        }  
  
        void setDataModel(DataModel model) {  
            //Update your UI with the data model passed here...  
        }  
    }  
}
```

```

text.setText(model.getText());
    checkBox.setChecked(model.isChecked());
}

public MyAdapter() {
mSortedList = new SortedList<>(DataModel.class, new
SortedListAdapterCallback<DataModel>(this) {
    @Override
    public int compare(DataModel o1, DataModel o2) {
        //此方法用于确定数组中对象的排序顺序。
        if (o1.someValue() < o2.someValue()) {
            return -1;
        } else if (o1.someValue() > o2.someValue()) {
            return 1;
        } else {
            return 0;
        }
    }

    @Override
    public boolean areContentsTheSame(DataModel oldItem, DataModel newItem) {
        //这是用来判断该对象的内容是否发生了变化。只有当 areItemsTheSame() 返回 true 时，这些项目才
被视为相等。
        //如果返回 false，则会调用 onBindViewHolder()，并传入包含该项目及其位置的 holder。
        return oldItem.getText().equals(newItem.getText()) && oldItem.isChecked() ==
newItem.isChecked();
    }

    @Override
    public boolean areItemsTheSame(DataModel item1, DataModel item2) {
        //检查这两个项目是否相同。如果不同，则添加到列表中，否则检查内容是否发生变化。
        return item1.equals(item2);
    }
});

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = //在这里初始化你的项目视图。
    return new ViewHolder(itemView);
}

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    //仅用给定位置的排序列表中的对象更新holder
    DataModel model = mSortedList.get(position);
    if (model != null) {
holder.setDataModel(model);
    }
}

@Override
public int getItemCount() {
    return mSortedList.size();
}

public void resetList(List<DataModel> models) {
    //如果执行多个更改，使用批处理方法以确保正确的
}

```

```

text.setText(module.getText());
checkBox.setChecked(model.isChecked());
}

public MyAdapter() {
mSortedList = new SortedList<>(DataModel.class, new
SortedListAdapterCallback<DataModel>(this) {
    @Override
    public int compare(DataModel o1, DataModel o2) {
        //This gets called to find the ordering between objects in the array.
        if (o1.someValue() < o2.someValue()) {
            return -1;
        } else if (o1.someValue() > o2.someValue()) {
            return 1;
        } else {
            return 0;
        }
    }

    @Override
    public boolean areContentsTheSame(DataModel oldItem, DataModel newItem) {
        //This is to see if the content of this object has changed. These items are only
considered equal if areItemsTheSame() returned true.
        //If this returns false, onBindViewHolder() is called with the holder containing the
item, and the item's position.
        return oldItem.getText().equals(newItem.getText()) && oldItem.isChecked() ==
newItem.isChecked();
    }

    @Override
    public boolean areItemsTheSame(DataModel item1, DataModel item2) {
        //Checks to see if these two items are the same. If not, it is added to the list,
otherwise, check if content has changed.
        return item1.equals(item2);
    }
};

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = //Initiate your item view here.
    return new ViewHolder(itemView);
}

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    //Just update the holder with the object in the sorted list from the given position
    DataModel model = mSortedList.get(position);
    if (model != null) {
        holder.setDataModel(model);
    }
}

@Override
public int getItemCount() {
    return mSortedList.size();
}

public void resetList(List<DataModel> models) {
    //If you are performing multiple changes, use the batching methods to ensure proper
}

```

```
动画。  
mSortedList.beginBatchedUpdates();  
    mSortedList.clear();  
mSortedList.addAll(models);  
    mSortedList.endBatchedUpdates();  
}  
  
//以下方法各自修改数据集，并自动处理调用适配器上的相应“notify”方法。
```

```
public void addModel(DataModel model) {  
    mSortedList.add(model);  
}  
  
public void addModels(List<DataModel> models) {  
    mSortedList.addAll(models);  
}  
  
public void clear() {  
    mSortedList.clear();  
}  
  
public void removeModel(DataModel model) {  
    mSortedList.remove(model);  
}  
  
public void removeModelAt(int i) {  
    mSortedList.removeItemAt(i);  
}
```

```
animation.  
mSortedList.beginBatchedUpdates();  
    mSortedList.clear();  
mSortedList.addAll(models);  
    mSortedList.endBatchedUpdates();  
}  
  
//The following methods each modify the data set and automatically handles calling the  
appropriate 'notify' method on the adapter.  
public void addModel(DataModel model) {  
    mSortedList.add(model);  
}  
  
public void addModels(List<DataModel> models) {  
    mSortedList.addAll(models);  
}  
  
public void clear() {  
    mSortedList.clear();  
}  
  
public void removeModel(DataModel model) {  
    mSortedList.remove(model);  
}  
  
public void removeModelAt(int i) {  
    mSortedList.removeItemAt(i);  
}
```

第16.5节：带有RecyclerView的弹出菜单

将此代码放入你的ViewHolder中

注意：在此代码中，我使用了btnExpand的点击事件，若要为整个RecyclerView设置点击事件，可以为itemView对象设置监听器。

```
public class MyViewHolder extends RecyclerView.ViewHolder{  
    CardView cv;  
    TextView recordName, visibleFile, date, time;  
    Button btnIn, btnExpand;  
  
    public MyViewHolder(final View itemView) {  
        super(itemView);  
  
        cv = (CardView)itemView.findViewById(R.id.cardview);  
        recordName = (TextView)itemView.findViewById(R.id.tv_record);  
        visibleFile = (TextView)itemView.findViewById(R.id.visible_file);  
        date = (TextView)itemView.findViewById(R.id.date);  
        time = (TextView)itemView.findViewById(R.id.time);  
        btnIn = (Button)itemView.findViewById(R.id.btn_in_out);  
  
        btnExpand = (Button) itemView.findViewById(R.id.btn_expand);  
  
        btnExpand.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);  
  
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
                    @Override
```

Section 16.5: Popup menu with recyclerView

put this code inside your ViewHolder

note: In this code I am using btnExpand click-event, for whole recyclerView click event you can set listener to itemView object.

```
public class MyViewHolder extends RecyclerView.ViewHolder{  
    CardView cv;  
    TextView recordName, visibleFile, date, time;  
    Button btnIn, btnExpand;  
  
    public MyViewHolder(final View itemView) {  
        super(itemView);  
  
        cv = (CardView)itemView.findViewById(R.id.cardview);  
        recordName = (TextView)itemView.findViewById(R.id.tv_record);  
        visibleFile = (TextView)itemView.findViewById(R.id.visible_file);  
        date = (TextView)itemView.findViewById(R.id.date);  
        time = (TextView)itemView.findViewById(R.id.time);  
        btnIn = (Button)itemView.findViewById(R.id.btn_in_out);  
  
        btnExpand = (Button) itemView.findViewById(R.id.btn_expand);  
  
        btnExpand.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);  
  
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
                    @Override
```

```

public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_delete:
            moveFile(recordName.getText().toString(),
getAdapterPosition());
            return true;
        case R.id.action_play:
            String valueOfPath = recordName.getText().toString();
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            File file = new File(valueOfPath);
            intent.setDataAndType(Uri.fromFile(file), "audio/*");
            context.startActivity(intent);
            return true;
        case R.id.action_share:
            String valueOfPath = recordName.getText().toString();
            File filee = new File(valueOfPath);
            try {
Intent sendIntent = new Intent();
                sendIntent.setAction(Intent.ACTION_SEND);
                sendIntent.setType("audio/*");
sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
context.startActivity(sendIntent);
            } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
e.printStackTrace();
            } catch (Exception e) {
}
            return true;
        default:
            return false;
    }
}
// 这里你可以加载你的菜单
popup.inflate(R.menu.my_menu_item);
popup.setGravity(Gravity.RIGHT);

// 如果你想在菜单项中显示图标, 请写这个try-catch块。
try {
    Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
    mFieldPopup.setAccessible(true);
    MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
    mPopup.setForceShowIcon(true);
} catch (Exception e) {

}
popup.show();
});
}
}

```

显示菜单图标的另一种方法

```

try {
    Field[] fields = popup.getClass().getDeclaredFields();
    for (Field field : fields) {

```

```

public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_delete:
            moveFile(recordName.getText().toString(),
getAdapterPosition());
            return true;
        case R.id.action_play:
            String valueOfPath = recordName.getText().toString();
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            File file = new File(valueOfPath);
            intent.setDataAndType(Uri.fromFile(file), "audio/*");
            context.startActivity(intent);
            return true;
        case R.id.action_share:
            String valueOfPath = recordName.getText().toString();
            File filee = new File(valueOfPath);
            try {
Intent sendIntent = new Intent();
                sendIntent.setAction(Intent.ACTION_SEND);
                sendIntent.setType("audio/*");
sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
context.startActivity(sendIntent);
            } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
e.printStackTrace();
            } catch (Exception e) {
}
            return true;
        default:
            return false;
    }
}
// here you can inflate your menu
popup.inflate(R.menu.my_menu_item);
popup.setGravity(Gravity.RIGHT);

// if you want icon with menu items then write this try-catch block.
try {
    Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
    mFieldPopup.setAccessible(true);
    MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
    mPopup.setForceShowIcon(true);
} catch (Exception e) {

}
popup.show();
});
}
}

```

alternative way to show icons in menu

```

try {
    Field[] fields = popup.getClass().getDeclaredFields();
    for (Field field : fields) {

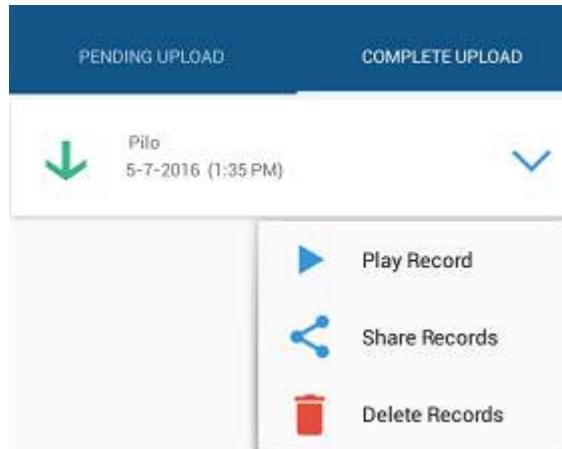
```

```

if ("mPopup".equals(field.getName())) {
    field.setAccessible(true);
    Object menuPopupHelper = field.get(menu);
    Class<?> classPopupHelper = Class.forName(menuPopupHelper
        .getClass().getName());
    Method setForceIcons = classPopupHelper.getMethod(
        "setForceShowIcon", boolean.class);
    setForceIcons.invoke(menuPopupHelper, true);
    break;
}
} catch (Exception e) {
}

```

输出如下：

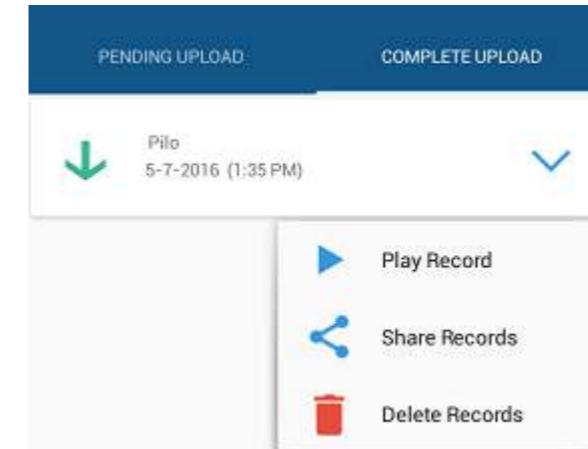


```

if ("mPopup".equals(field.getName())) {
    field.setAccessible(true);
    Object menuPopupHelper = field.get(menu);
    Class<?> classPopupHelper = Class.forName(menuPopupHelper
        .getClass().getName());
    Method setForceIcons = classPopupHelper.getMethod(
        "setForceShowIcon", boolean.class);
    setForceIcons.invoke(menuPopupHelper, true);
    break;
}
} catch (Exception e) {
}

```

Here is the output:



第16.6节：使用多个带有ItemViewType的ViewHolder

有时RecyclerView需要使用多种类型的视图来显示在UI中的列表中，且每个视图需要不同的布局xml进行加载。

针对这个问题，可以在单个Adapter中使用不同的ViewHolder，通过RecyclerView中的一个特殊方法 -
getItemViewType(int position)。

下面是使用两个ViewHolder的示例：

1.用于显示列表条目的ViewHolder

2.用于显示多个头部视图的ViewHolder

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(context).inflate(viewType, parent, false);
    return ViewHolder.create(itemView, viewType);
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    final Item model = this.items.get(position);
    ((ViewHolder) holder).bind(model);
}

@Override
public int getItemViewType(int position) {
}

```

Section 16.6: Using several ViewHolders with ItemViewType

Sometimes a RecyclerView will need to use several types of Views to be displayed in the list shown in the UI, and each View needs a different layout xml to be inflated.

For this issue, you may use different ViewHolders in single Adapter, by using a special method in RecyclerView -
getItemViewType(int position).

Below is example of using two ViewHolders:

1. A ViewHolder for displaying list entries
2. A ViewHolder for displaying multiple header views

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(context).inflate(viewType, parent, false);
    return ViewHolder.create(itemView, viewType);
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    final Item model = this.items.get(position);
    ((ViewHolder) holder).bind(model);
}

@Override
public int getItemViewType(int position) {
}

```

```

        return inSearchState ? R.layout.item_header : R.layout.item_entry;
    }

    abstract class ViewHolder {
        abstract void bind(Item model);

        public static ViewHolder create(View v, int viewType) {
            return viewType == R.layout.item_header ? new HeaderViewHolder(v) :new
EntryViewHolder(v);
        }
    }

    static class EntryViewHolder extends ViewHolder {
        private View v;

        public EntryViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // 将项目数据绑定到条目视图。
        }
    }

    static class HeaderViewHolder extends ViewHolder {
        private View v;

        public HeaderViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // 将项目数据绑定到头部视图。
        }
    }
}

```

```

        return inSearchState ? R.layout.item_header : R.layout.item_entry;
    }

    abstract class ViewHolder {
        abstract void bind(Item model);

        public static ViewHolder create(View v, int viewType) {
            return viewType == R.layout.item_header ? new HeaderViewHolder(v) :new
EntryViewHolder(v);
        }
    }

    static class EntryViewHolder extends ViewHolder {
        private View v;

        public EntryViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // Bind item data to entry view.
        }
    }

    static class HeaderViewHolder extends ViewHolder {
        private View v;

        public HeaderViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // Bind item data to header view.
        }
    }
}

```

第16.7节：使用

SearchView过滤RecyclerView中的项目

在RecyclerView.Adapter中添加filter方法：

```

public void filter(String text) {
    if(text.isEmpty()){
        items.clear();
        items.addAll(itemsCopy);
    } else{
        ArrayList<PhoneBookItem> result = new ArrayList<>();
        text = text.toLowerCase();
        for(PhoneBookItem item: itemsCopy){
            //按姓名或电话匹配
            if(item.name.toLowerCase().contains(text) ||
item.phone.toLowerCase().contains(text)){
                result.add(item);
            }
        }
        items.clear();
        items.addAll(result);
    }
    notifyDataSetChanged();
}

```

Section 16.7: Filter items inside RecyclerView with a SearchView

add filter method in RecyclerView.Adapter:

```

public void filter(String text) {
    if(text.isEmpty()){
        items.clear();
        items.addAll(itemsCopy);
    } else{
        ArrayList<PhoneBookItem> result = new ArrayList<>();
        text = text.toLowerCase();
        for(PhoneBookItem item: itemsCopy){
            //match by name or phone
            if(item.name.toLowerCase().contains(text) ||
item.phone.toLowerCase().contains(text)){
                result.add(item);
            }
        }
        items.clear();
        items.addAll(result);
    }
    notifyDataSetChanged();
}

```

```
}
```

itemsCopy 在适配器的构造函数中初始化，如itemsCopy.addAll(items)。

如果是这样，只需在SearchView的OnQueryTextListener中调用filter：

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        adapter.filter(query);
        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.filter(newText);
        return true;
    }
});
```

第16.8节：RecyclerView的拖拽和滑动

您可以使用RecyclerView实现滑动删除和拖放功能，而无需使用第三方库。

只需使用 RecyclerView 支持库中包含的 ItemTouchHelper 类。

使用 SimpleCallback 回调实例化 ItemTouchHelper，根据你支持的功能，应该重写 onMove(RecyclerView, ViewHolder, ViewHolder) 和/或 onSwiped(ViewHolder,int)，最后将其附加到你的 RecyclerView 上。

```
ItemTouchHelper.SimpleCallback simpleItemTouchCallback = new ItemTouchHelper.SimpleCallback(0,
    ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
        // 从适配器中移除项目
    }

    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
        RecyclerView.ViewHolder target) {
        final int fromPos = viewHolder.getAdapterPosition();
        final int toPos = target.getAdapterPosition();
        // 在适配器中将项目从 `fromPos` 移动到 `toPos`。
        return true; // 移动成功返回 true，否则返回 false
    }
};

ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleItemTouchCallback);
itemTouchHelper.attachToRecyclerView(recyclerView);
```

值得一提的是，SimpleCallback 构造函数对 RecyclerView 中的所有项目应用相同的滑动策略。无论如何，可以通过简单地重写方法 getSwipeDirs(RecyclerView, ViewHolder) 来更新特定项目的默认滑动方向。

假设例如我们的RecyclerView包含一个HeaderViewHolder，显然我们不想对它应用滑动操作。只需重写getSwipeDirs方法，如下所示：

```
}
```

itemsCopy 是初始化在适配器的构造函数中 like itemsCopy.addAll(items)。

If you do so, just call filter from OnQueryTextListener from SearchView:

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        adapter.filter(query);
        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.filter(newText);
        return true;
    }
});
```

Section 16.8: Drag&Drop and Swipe with RecyclerView

You can implement the swipe-to-dismiss and drag-and-drop features with the RecyclerView without using 3rd party libraries.

Just use the [ItemTouchHelper](#) class included in the RecyclerView support library.

Instantiate the ItemTouchHelper with the [SimpleCallback](#) callback and depending on which functionality you support, you should override onMove(RecyclerView, ViewHolder, ViewHolder) and / or onSwiped(ViewHolder, int) and finally attach to your RecyclerView.

```
ItemTouchHelper.SimpleCallback simpleItemTouchCallback = new ItemTouchHelper.SimpleCallback(0,
    ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
        // remove item from adapter
    }

    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
        RecyclerView.ViewHolder target) {
        final int fromPos = viewHolder.getAdapterPosition();
        final int toPos = target.getAdapterPosition();
        // move item in `fromPos` to `toPos` in adapter.
        return true; // true if moved, false otherwise
    }
};

ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleItemTouchCallback);
itemTouchHelper.attachToRecyclerView(recyclerView);
```

It's worth mentioning that SimpleCallback constructor applies the same swiping strategy to all items in the RecyclerView. It's possible in any case to update the default swiping direction for specific items by simply overriding method getSwipeDirs(RecyclerView, ViewHolder).

Let's suppose for example that our RecyclerView includes a HeaderViewHolder and that we obviously don't want to apply swiping to it. It will be enough to override getSwipeDirs as follows:

```

@Override
public int getSwipeDirs(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder) {
    if (viewHolder instanceof HeaderViewHolder) {
        // 头部不允许滑动
        return 0;
    }
    // 其他所有项默认允许滑动
    return super.getSwipeDirs(recyclerView, viewHolder);
}

```

第16.9节：在项目加载或数据不可用时显示默认视图

截图



适配器类

```

private class MyAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    final int EMPTY_VIEW = 77777;
    List<CustomData> dataList = new ArrayList<>();

    MyAdapter() {
        super();
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflator layoutInflater = LayoutInflator.from(parent.getContext());

```

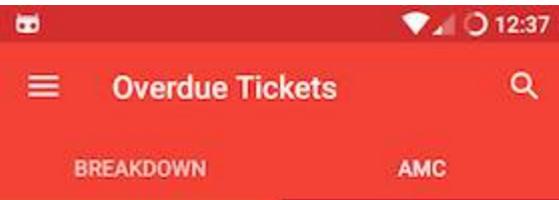
```

@Override
public int getSwipeDirs(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder) {
    if (viewHolder instanceof HeaderViewHolder) {
        // no swipe for header
        return 0;
    }
    // default swipe for all other items
    return super.getSwipeDirs(recyclerView, viewHolder);
}

```

Section 16.9: Show default view till items load or when data is not available

Screenshot



Adapter Class

```

private class MyAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    final int EMPTY_VIEW = 77777;
    List<CustomData> dataList = new ArrayList<>();

    MyAdapter() {
        super();
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflator layoutInflater = LayoutInflator.from(parent.getContext());

```

```

if (viewType == EMPTY_VIEW) {
    return new EmptyView(layoutInflater.inflate(R.layout.nothing_yet, parent, false));
} else {
    return new ItemView(layoutInflater.inflate(R.layout.my_item, parent, false));
}

@SuppressLint("SetTextI18n")
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {
    if (getItemViewType(position) == EMPTY_VIEW) {
        EmptyView emptyView = (EmptyView) holder;
        emptyView.primaryText.setText("暂无数据");
        emptyView.secondaryText.setText("你做得很好！");
        emptyView.primaryText.setCompoundDrawablesWithIntrinsicBounds(null, new
IconicsDrawable(getActivity()).icon(FontAwesome.Icon.faw_ticket).sizeDp(48).color(Color.DKGRAY),
null, null);
    } 否则 {
        ItemView itemView = (ItemView) holder;
        // 绑定数据到 itemView
    }
}

@Override
public int getItemCount() {
    return dataList.size() > 0 ? dataList.size() : 1;
}

@Override
public int getItemViewType(int position) {
    if (dataList.size() == 0) {
        return EMPTY_VIEW;
    }
    return super.getItemViewType(position);
}
}

```

nothing_yet.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:orientation="vertical"
    android:paddingBottom="100dp"
    android:paddingTop="100dp">

    <TextView
        android:id="@+id/nothingPrimary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:drawableTint="@android:color/secondary_text_light"
        android:drawableTop="@drawable/ic_folder_open_black_24dp"
        android:enabled="false"
        android:fontFamily="sans-serif-light"
        android:text="暂无项目"/>

```

```

if (viewType == EMPTY_VIEW) {
    return new EmptyView(layoutInflater.inflate(R.layout.nothing_yet, parent, false));
} else {
    return new ItemView(layoutInflater.inflate(R.layout.my_item, parent, false));
}

@SuppressLint("SetTextI18n")
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {
    if (getItemViewType(position) == EMPTY_VIEW) {
        EmptyView emptyView = (EmptyView) holder;
        emptyView.primaryText.setText("No data yet");
        emptyView.secondaryText.setText("You're doing good !");
        emptyView.primaryText.setCompoundDrawablesWithIntrinsicBounds(null, new
IconicsDrawable(getActivity()).icon(FontAwesome.Icon.faw_ticket).sizeDp(48).color(Color.DKGRAY),
null, null);
    } else {
        ItemView itemView = (ItemView) holder;
        // Bind data to itemView
    }
}

@Override
public int getItemCount() {
    return dataList.size() > 0 ? dataList.size() : 1;
}

@Override
public int getItemViewType(int position) {
    if (dataList.size() == 0) {
        return EMPTY_VIEW;
    }
    return super.getItemViewType(position);
}
}

```

nothing_yet.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:orientation="vertical"
    android:paddingBottom="100dp"
    android:paddingTop="100dp">

    <TextView
        android:id="@+id/nothingPrimary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:drawableTint="@android:color/secondary_text_light"
        android:drawableTop="@drawable/ic_folder_open_black_24dp"
        android:enabled="false"
        android:fontFamily="sans-serif-light"
        android:text="No Item's Yet"/>

```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@android:color/secondary_text_light"
        android:textSize="40sp"
        tools:targetApi="m" />
```

```
<TextView
    android:id="@+id/nothingSecondary"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:enabled="false"
    android:fontFamily="sans-serif-condensed"
    android:text="你做得很好！"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:textColor="@android:color/tertiary_text_light" />
</LinearLayout>
```

我正在使用 FontAwesome 和 Iconics 库来显示图标。将此添加到你的应用级 build.gradle 文件中。

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

第16.10节：为 RecyclerView 添加头部/尾部

这是一个示例适配器代码。

```
public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int FOOTER_VIEW = 1;

    // 定义Footer视图的ViewHolder

    public class FooterViewHolder extends ViewHolder {
        public FooterViewHolder(View itemView) {
            super(itemView);
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 点击该项时执行你想做的操作
                }
            });
        }
    }

    // 现在定义普通列表项的ViewHolder
    public class NormalViewHolder extends ViewHolder {
        public NormalViewHolder(View itemView) {
            super(itemView);

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 点击普通项时执行你想做的操作
                }
            });
        }
    }

    // 现在在onCreateViewHolder中，你需要传入正确的视图
    // 用于填充列表项。
}
```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@android:color/secondary_text_light"
        android:textSize="40sp"
        tools:targetApi="m" />
```

```
<TextView
    android:id="@+id/nothingSecondary"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:enabled="false"
    android:fontFamily="sans-serif-condensed"
    android:text="You're doing good !"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:textColor="@android:color/tertiary_text_light" />
</LinearLayout>
```

I'm using FontAwesome with Iconics Library for the images. Add this to your app level build.gradle file.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Section 16.10: Add header/footer to a RecyclerView

This is a sample adapter code.

```
public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int FOOTER_VIEW = 1;

    // Define a view holder for Footer view

    public class FooterViewHolder extends ViewHolder {
        public FooterViewHolder(View itemView) {
            super(itemView);
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // Do whatever you want on clicking the item
                }
            });
        }
    }

    // Now define the viewholder for Normal list item
    public class NormalViewHolder extends ViewHolder {
        public NormalViewHolder(View itemView) {
            super(itemView);

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // Do whatever you want on clicking the normal items
                }
            });
        }
    }

    // And now in onCreateViewHolder you have to pass the correct view
    // while populating the list item.
}
```

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    查看 v;

    如果 (viewType == FOOTER_VIEW) {
        v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_footer, parent,
        false);
    }

    FooterViewHolder vh = new FooterViewHolder(v);

    返回 vh;
}

v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_normal, parent, false);
NormalViewHolder vh = new NormalViewHolder(v);

返回 vh;
}

// 现在在 onBindViewHolder 中绑定 viewholders
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {

    try {
        如果 (holder instanceof NormalViewHolder) {
            NormalViewHolder vh = (NormalViewHolder) holder;

            vh.bindView(position);
        } else if (holder instanceof FooterViewHolder) {
            FooterViewHolder vh = (FooterViewHolder) holder;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// 现在是关键部分。你必须返回列表的准确项目数量
// 我这里只有一个底部视图。所以我返回了 data.size() + 1
// 如果你有多个头部和底部视图，你需要返回总数
// 比如，headers.size() + data.size() + footers.size()

@Override
public int getItemCount() {
    if (data == null) {
        return 0;
    }

    if (data.size() == 0) {
        // 这里返回 1 表示显示空内容
        return 1;
    }

    // 添加额外视图以显示底部视图
    return data.size() + 1;
}

// 现在定义你自己的 getItemViewType。

@Override
public int getItemViewType(int position) {

```

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v;

    if (viewType == FOOTER_VIEW) {
        v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_footer, parent,
        false);
    }

    FooterViewHolder vh = new FooterViewHolder(v);

    return vh;
}

v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_normal, parent, false);
NormalViewHolder vh = new NormalViewHolder(v);

return vh;
}

// Now bind the viewholders in onBindViewHolder
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {

    try {
        if (holder instanceof NormalViewHolder) {
            NormalViewHolder vh = (NormalViewHolder) holder;

            vh.bindView(position);
        } else if (holder instanceof FooterViewHolder) {
            FooterViewHolder vh = (FooterViewHolder) holder;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Now the critical part. You have return the exact item count of your list
// I've only one footer. So I returned data.size() + 1
// If you've multiple headers and footers, you've to return total count
// like, headers.size() + data.size() + footers.size()

@Override
public int getItemCount() {
    if (data == null) {
        return 0;
    }

    if (data.size() == 0) {
        //Return 1 here to show nothing
        return 1;
    }

    // Add extra view to show the footer view
    return data.size() + 1;
}

// Now define getItemViewType of your own.

@Override
public int getItemViewType(int position) {

```

```

if (position == data.size()) {
    // 这里我们将添加页脚。
    return FOOTER_VIEW;
}

return super.getItemViewType(position);
}

// 所以你已经完成了添加页脚及其 onClick 操作。
// 现在为 NormalViewHolder 设置默认的 ViewHolder

public class ViewHolder extends RecyclerView.ViewHolder {
    // 在这里定义一行的元素
    public ViewHolder(View itemView) {
        super(itemView);
        // 通过 ID 查找视图并在这里初始化
    }

    public void bindView(int position) {
        // 实现操作的 bindView() 方法
    }
}

```

这里有
一篇关
于实
现

带有头部和尾部的RecyclerView。

替代方法：

虽然上述答案可行，但你也可以使用这种方法，在NestedScrollView中使用RecyclerView。你可以通过以下方法添加头部布局：

```

<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <include
            layout="@layout/drawer_view_header"
            android:id="@+id/navigation_header" />

        <android.support.v7.widget.RecyclerView
            android:layout_below="@+id/navigation_header"
            android:id="@+id/followers_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </RelativeLayout>
</android.support.v4.widget.NestedScrollView>

```

或者你也可以在NestedScrollView中使用垂直排列的LinearLayout。

注意：这只适用于版本高于23.2.0的RecyclerView

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

```

if (position == data.size()) {
    // This is where we'll add footer.
    return FOOTER_VIEW;
}

return super.getItemViewType(position);
}

// So you're done with adding a footer and its action on onClick.
// Now set the default ViewHolder for NormalViewHolder

public class ViewHolder extends RecyclerView.ViewHolder {
    // Define elements of a row here
    public ViewHolder(View itemView) {
        super(itemView);
        // Find view by ID and initialize here
    }

    public void bindView(int position) {
        // bindView() method to implement actions
    }
}

```

[Here's a good read](#) about the implementation of RecyclerView with header and footer.

Alternate method:

While the above answer will work you can use this approach as well using a recycler view using a NestedScrollView. You can add a layout for header using the following approach:

```

<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <include
            layout="@layout/drawer_view_header"
            android:id="@+id/navigation_header" />

        <android.support.v7.widget.RecyclerView
            android:layout_below="@+id/navigation_header"
            android:id="@+id/followers_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </RelativeLayout>
</android.support.v4.widget.NestedScrollView>

```

Or you may also use a LinearLayout with vertical alignment in your NestedScrollView.

Note: This will only work with RecyclerView above 23.2.0

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

第16.11节：RecyclerView中的无限滚动

这里我分享了一个在RecyclerView中实现无限滚动的代码片段。

步骤1：首先在RecyclerView适配器中创建一个抽象方法，如下所示。

```
public abstract class ViewAllCategoryAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>
{
    public abstract void load();
}
```

步骤2：现在重写ViewAllCategoryAdapter类的onBindViewHolder和getItemCount()方法，并像下面这样调用load()方法。

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

步骤3：现在所有后端逻辑都完成了，接下来是执行这段逻辑。很简单，你可以重写load方法，在那里创建你的适配器对象。当用户滚动到列表末尾时，这个方法会自动调用。

```
adapter = new ViewAllCategoryAdapter(CONTEXT, YOURLIST) {
    @Override
    public void load() {

        /* 在这里执行你的操作 */
        /* 当用户滚动到列表末尾时，此方法会自动调用。*/
    }
};
recycleCategory.setAdapter(adapter);
```

现在，当用户滚动到列表末尾时，load()方法会自动调用。

祝你好运

第16.12节：为RecyclerView项目添加分割线

只需将以下代码添加到初始化中

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
    DividerItemDecoration.VERTICAL));
```

添加一个适配器并像往常一样调用.notifyDataSetChanged();

这不是RecyclerView的内置功能，而是添加在支持库中的功能。所以别忘了在你的应用级build.gradle文件中包含它

Section 16.11: Endless Scrolling in Recycleview

Here I have shared a code snippet for implementing endless scrolling in recycle view.

Step 1: First make a one abstract method in Recycleview adapter like below.

```
public abstract class ViewAllCategoryAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>
{
    public abstract void load();
}
```

Step 2: Now override [onBindViewHolder](#) and [getItemCount\(\)](#) method of ViewAllCategoryAdapter class and call Load() method like below.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

Step 3: Now every backend logic is complete now it's time to execute this logic. It's simple you can override load method where you create object of your adapter. this method is automatically call while user reach at end of the listing.

```
adapter = new ViewAllCategoryAdapter(CONTEXT, YOURLIST) {
    @Override
    public void load() {

        /* do your stuff here */
        /* This method is automatically call while user reach at end of your list. */
    }
};
recycleCategory.setAdapter(adapter);
```

Now load() method automatically call while user scroll at end of list.

Best Luck

Section 16.12: Add divider lines to RecyclerView items

Just add these lines to the initialization

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
    DividerItemDecoration.VERTICAL));
```

Add an adapter and call [.notifyDataSetChanged\(\)](#) ; as usual !

This is not an inbuilt feature of Recyclerview but added in the support libraries. So don't forget to include this in your app level build.gradle file

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

可以向单个RecyclerView添加多个ItemDecoration。

更改分割线颜色：

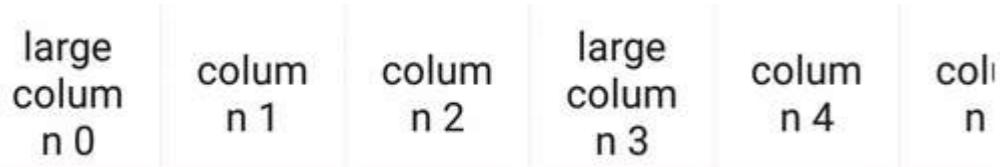
为itemDecoration设置颜色非常简单。

1. 步骤是：创建一个divider.xml文件，位于drawable文件夹中

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

第二步是：设置drawable

```
// 获取drawable对象
Drawable mDivider = ContextCompat.getDrawable(m_jContext, R.drawable.divider);
// 创建一个方向为水平的DividerItemDecoration
DividerItemDecoration hItemDecoration = new DividerItemDecoration(m_jContext,
    DividerItemDecoration.HORIZONTAL);
// 在其上设置drawable
hItemDecoration.setDrawable(mDivider);
```



```
// 创建一个方向为垂直的DividerItemDecoration
DividerItemDecoration vItemDecoration = new DividerItemDecoration(m_jContext,
    DividerItemDecoration.VERTICAL);
// 在其上设置drawable
vItemDecoration.setDrawable(mDivider);
```

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

Multiple ItemDecorations can be added to a single RecyclerView.

Changing divider color :

It's pretty easy to set a color for a itemDecoration.

1. step is: creating a divider.xml file which is located on drawable folder

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. step is: setting drawable

```
// Get drawable object
Drawable mDivider = ContextCompat.getDrawable(m_jContext, R.drawable.divider);
// Create a DividerItemDecoration whose orientation is Horizontal
DividerItemDecoration hItemDecoration = new DividerItemDecoration(m_jContext,
    DividerItemDecoration.HORIZONTAL);
// Set the drawable on it
hItemDecoration.setDrawable(mDivider);
```



```
// Create a DividerItemDecoration whose orientation is vertical
DividerItemDecoration vItemDecoration = new DividerItemDecoration(m_jContext,
    DividerItemDecoration.VERTICAL);
// Set the drawable on it
vItemDecoration.setDrawable(mDivider);
```

row 7

row 8

row 9

row 10

row 11

row 12

row 13

row 7

row 8

row 9

row 10

row 11

row 12

row 13

第17章：RecyclerView装饰

参数

装饰 要添加到RecyclerView的项目装饰

该RecyclerView装饰列表中的索引。这是调用getItemOffset和onDraw的顺序。后调用的可能会覆盖之前的。

详情

第17.1节：为RecyclerView添加分割线

首先你需要创建一个继承自RecyclerView.ItemDecoration的类：

```
public class SimpleBlueDivider extends RecyclerView.ItemDecoration {  
    private Drawable mDivider;  
  
    public SimpleBlueDivider(Context context) {  
        mDivider = context.getResources().getDrawable(R.drawable.divider_blue);  
    }  
  
    @Override  
    public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {  
        //分割线内边距，给一些你想要的内边距或禁用  
        int left = parent.getPaddingLeft() + 80;  
        int right = parent.getWidth() - parent.getPaddingRight() - 30;  
  
        int childCount = parent.getChildCount();  
        for (int i = 0; i < childCount; i++) {  
            View child = parent.getChildAt(i);  
  
            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();  
  
            int top = child.getBottom() + params.bottomMargin;  
            int bottom = top + mDivider.getIntrinsicHeight();  
  
            mDivider.setBounds(left, top, right, bottom);  
            mDivider.draw(c);  
        }  
    }  
}
```

将 divider_blue.xml 添加到你的 drawable 文件夹：

```
<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">  
    <size android:width="1dp" android:height="4dp" />  
    <solid android:color="#AA123456" />  
</shape>
```

然后像这样使用：

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

结果将会是：

Chapter 17: RecyclerView Decorations

Parameter

decoration the item decoration to add to the RecyclerView

index the index in the list of decorations for this RecyclerView. This is the order in which getItemOffset and onDraw are called. Later calls might overdraw previous ones.

Details

First of all you need to create a class which extends RecyclerView.ItemDecoration :

```
public class SimpleBlueDivider extends RecyclerView.ItemDecoration {  
    private Drawable mDivider;  
  
    public SimpleBlueDivider(Context context) {  
        mDivider = context.getResources().getDrawable(R.drawable.divider_blue);  
    }  
  
    @Override  
    public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {  
        //divider padding give some padding whatever u want or disable  
        int left = parent.getPaddingLeft() + 80;  
        int right = parent.getWidth() - parent.getPaddingRight() - 30;  
  
        int childCount = parent.getChildCount();  
        for (int i = 0; i < childCount; i++) {  
            View child = parent.getChildAt(i);  
  
            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();  
  
            int top = child.getBottom() + params.bottomMargin;  
            int bottom = top + mDivider.getIntrinsicHeight();  
  
            mDivider.setBounds(left, top, right, bottom);  
            mDivider.draw(c);  
        }  
    }  
}
```

Add divider_blue.xml to your drawable folder :

```
<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">  
    <size android:width="1dp" android:height="4dp" />  
    <solid android:color="#AA123456" />  
</shape>
```

Then use it like :

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

The result will be like :



Stackoverflow :)

这张图片只是一个示例，展示了分隔线的工作原理，如果你想在添加分隔线时遵循Material Design规范，请查看这个链接：[dividers](#)，感谢@Brenden Kromhout提供链接。



Stackoverflow :)

This image is just an example how dividers working , if you want to follow Material Design specs when adding dividers please take a look at this link : [dividers](#) and thanks @Brenden Kromhout by providing link .

第17.2节：绘制分隔线

这将在每个视图的底部绘制一条线，但最后一个除外，用作项目之间的分隔线。

```
public class SeparatorDecoration extends RecyclerView.ItemDecoration {

    private final Paint mPaint;
    private final int mAlpha;

    public SeparatorDecoration(@ColorInt int color, float width) {
        mPaint = new Paint();
        mPaint.setColor(color);
        mPaint.setStrokeWidth(width);
        mAlpha = mPaint.getAlpha();
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.State state) {
        final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
        view.getLayoutParams();

        // 我们获取列表中的位置
        final int position = params.getViewAdapterPosition();

        // 为每个视图底部（除最后一个外）添加分隔符的空间
        if (position < state.getItemCount()) {
            outRect.set(0, 0, 0, (int) mPaint.getStrokeWidth()); // 左、上、右、下
        } else {
            outRect.setEmpty(); // 0, 0, 0, 0
        }
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        // 一条线会向上和向下各绘制一半,
        // 因此需要偏移以正确放置
        final int offset = (int) (mPaint.getStrokeWidth() / 2);

        // 这将遍历每个可见视图
        for (int i = 0; i < parent.getChildCount(); i++) {
            final View view = parent.getChildAt(i);
            final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
            view.getLayoutParams();

            // 获取位置
            final int position = params.getViewAdapterPosition();

            // 最后绘制分隔线
            if (position < state.getItemCount()) {
                // 应用透明度以支持动画效果
                mPaint.setAlpha((int) (view.getAlpha() * mAlpha));

                float positionY = view.getBottom() + offset + view.getTranslationY();
                // 执行绘制操作
                c.drawLine(view.getLeft() + view.getTranslationX(),
                          positionY,
                          view.getRight() + view.getTranslationX(),
                          positionY,
                          mPaint);
            }
        }
    }
}
```

Section 17.2: Drawing a Separator

This will draw a line at the bottom of every view but the last to act as a separator between items.

```
public class SeparatorDecoration extends RecyclerView.ItemDecoration {

    private final Paint mPaint;
    private final int mAlpha;

    public SeparatorDecoration(@ColorInt int color, float width) {
        mPaint = new Paint();
        mPaint.setColor(color);
        mPaint.setStrokeWidth(width);
        mAlpha = mPaint.getAlpha();
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.State state) {
        final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
        view.getLayoutParams();

        // we retrieve the position in the list
        final int position = params.getViewAdapterPosition();

        // add space for the separator to the bottom of every view but the last one
        if (position < state.getItemCount()) {
            outRect.set(0, 0, 0, (int) mPaint.getStrokeWidth()); // left, top, right, bottom
        } else {
            outRect.setEmpty(); // 0, 0, 0, 0
        }
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        // a line will draw half its size to top and bottom,
        // hence the offset to place it correctly
        final int offset = (int) (mPaint.getStrokeWidth() / 2);

        // this will iterate over every visible view
        for (int i = 0; i < parent.getChildCount(); i++) {
            final View view = parent.getChildAt(i);
            final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
            view.getLayoutParams();

            // get the position
            final int position = params.getViewAdapterPosition();

            // and finally draw the separator
            if (position < state.getItemCount()) {
                // apply alpha to support animations
                mPaint.setAlpha((int) (view.getAlpha() * mAlpha));

                float positionY = view.getBottom() + offset + view.getTranslationY();
                // do the drawing
                c.drawLine(view.getLeft() + view.getTranslationX(),
                          positionY,
                          view.getRight() + view.getTranslationX(),
                          positionY,
                          mPaint);
            }
        }
    }
}
```

```
        }  
    }  
}
```

第17.3节：如何使用DividerItemDecoration 添加分隔线

`DividerItemDecoration` 是一个 `RecyclerView.ItemDecoration`，可以用作项目之间的分隔线。

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,  
    mLayoutManager.getOrientation());  
recyclerView.addItemDecoration(mDividerItemDecoration);
```

它支持使用 `DividerItemDecoration.VERTICAL` 和 `DividerItemDecoration.HORIZONTAL` 两种方向。

第17.4节：使用 ItemDecoration 设置每个项目的边距

你可以使用 `RecyclerView.ItemDecoration` 在 `RecyclerView` 中的每个项目周围添加额外的边距。在某些情况下，这可以简化你的适配器实现和项目视图的 XML。

```
public class MyItemDecoration  
    extends RecyclerView.ItemDecoration {  
  
    private final int extraMargin;  
  
    @Override  
    public void getItemOffsets(Rect outRect, View view,  
        RecyclerView parent, RecyclerView.State state) {  
  
        int position = parent.getChildAdapterPosition(view);  
  
        // 给最后一项添加额外边距很容易...  
        if (position + 1 == parent.getAdapter().getItemCount()) {  
            outRect.bottom = extraMargin; // 单位是像素  
  
            // ...或者你可以根据RecyclerView中每个项目的位置  
            // 给它们设置不同的边距...  
            if (position % 2 == 0) {  
                outRect.right = extraMargin;  
            } else {  
                outRect.left = extraMargin;  
            }  
  
            // ...或者根据项目本身的某些属性  
            MyListItem item = parent.getAdapter().getItem(position);  
            if (item.isFirstItemInSection()) {  
                outRect.top = extraMargin;  
            }  
  
        }  
  
        public MyItemDecoration(Context context) {  
            extraMargin = context.getResources()  
                .getDimensionPixelOffset(R.dimen.extra_margin);  
        }  
    }
```

要启用装饰，只需将其添加到你的`RecyclerView`中：

```
        }  
    }  
}
```

Section 17.3: How to add dividers using and DividerItemDecoration

The `DividerItemDecoration` is a `RecyclerView.ItemDecoration` that can be used as a divider between items.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,  
    mLayoutManager.getOrientation());  
recyclerView.addItemDecoration(mDividerItemDecoration);
```

It supports both orientation using `DividerItemDecoration.VERTICAL` and `DividerItemDecoration.HORIZONTAL`.

Section 17.4: Per-item margins with ItemDecoration

You can use a `RecyclerView.ItemDecoration` to put extra margins around each item in a `RecyclerView`. This can in some cases clean up both your adapter implementation and your item view XML.

```
public class MyItemDecoration  
    extends RecyclerView.ItemDecoration {  
  
    private final int extraMargin;  
  
    @Override  
    public void getItemOffsets(Rect outRect, View view,  
        RecyclerView parent, RecyclerView.State state) {  
  
        int position = parent.getChildAdapterPosition(view);  
  
        // It's easy to put extra margin on the last item...  
        if (position + 1 == parent.getAdapter().getItemCount()) {  
            outRect.bottom = extraMargin; // unit is px  
        }  
  
        // ...or you could give each item in the RecyclerView different  
        // margins based on its position...  
        if (position % 2 == 0) {  
            outRect.right = extraMargin;  
        } else {  
            outRect.left = extraMargin;  
        }  
  
        // ...or based on some property of the item itself  
        MyListItem item = parent.getAdapter().getItem(position);  
        if (item.isFirstItemInSection()) {  
            outRect.top = extraMargin;  
        }  
  
    }  
  
    public MyItemDecoration(Context context) {  
        extraMargin = context.getResources()  
            .getDimensionPixelOffset(R.dimen.extra_margin);  
    }  
}
```

To enable the decoration, simply add it to your `RecyclerView`:

```
// 在你的onCreate()中  
RecyclerView rv = (RecyclerView) findViewById(R.id myList);  
rv.addItemDecoration(new MyItemDecoration(context));
```

第17.5节：RecyclerView中GridLayoutManager的ItemOffsetDecoration

以下示例将帮助为GridLayout中的项目提供相等的距离。

ItemOffsetDecoration.java

```
public class ItemOffsetDecoration extends RecyclerView.ItemDecoration {  
  
    private int mItemOffset;  
  
    private int spanCount = 2;  
  
    public ItemOffsetDecoration(int itemOffset) {  
        mItemOffset = itemOffset;  
    }  
  
    public ItemOffsetDecoration(@NonNull Context context, @DimenRes int itemOffsetId) {  
        this(context.getResources().getDimensionPixelSize(itemOffsetId));  
    }  
  
    @Override  
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,  
                               RecyclerView.State state) {  
        super.getItemOffsets(outRect, view, parent, state);  
  
        int position = parent.getChildLayoutPosition(view);  
  
        GridLayoutManager manager = (GridLayoutManager) parent.getLayoutManager();  
  
        if (position < manager.getSpanCount())  
            outRect.top = mItemOffset;  
  
        if (position % 2 != 0) {  
            outRect.right = mItemOffset;  
        }  
  
        outRect.left = mItemOffset;  
        outRect.bottom = mItemOffset;  
    }  
}
```

你可以像下面的代码那样调用 ItemDecoration。

```
recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);  
  
GridLayoutManager lLayout = new GridLayoutManager(getActivity(), 2);  
  
ItemOffsetDecoration itemDecoration = new ItemOffsetDecoration(mActivity, R.dimen.item_offset);  
recyclerView.addItemDecoration(itemDecoration);  
  
recyclerView.setLayoutManager(lLayout);
```

以及示例的项目偏移量

```
// in your onCreate()  
RecyclerView rv = (RecyclerView) findViewById(R.id myList);  
rv.addItemDecoration(new MyItemDecoration(context));
```

Section 17.5: ItemOffsetDecoration for GridLayoutManager in RecycleView

Following example will help to give equal space to an item in GridLayout.

ItemOffsetDecoration.java

```
public class ItemOffsetDecoration extends RecyclerView.ItemDecoration {  
  
    private int mItemOffset;  
  
    private int spanCount = 2;  
  
    public ItemOffsetDecoration(int itemOffset) {  
        mItemOffset = itemOffset;  
    }  
  
    public ItemOffsetDecoration(@NonNull Context context, @DimenRes int itemOffsetId) {  
        this(context.getResources().getDimensionPixelSize(itemOffsetId));  
    }  
  
    @Override  
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,  
                               RecyclerView.State state) {  
        super.getItemOffsets(outRect, view, parent, state);  
  
        int position = parent.getChildLayoutPosition(view);  
  
        GridLayoutManager manager = (GridLayoutManager) parent.getLayoutManager();  
  
        if (position < manager.getSpanCount())  
            outRect.top = mItemOffset;  
  
        if (position % 2 != 0) {  
            outRect.right = mItemOffset;  
        }  
  
        outRect.left = mItemOffset;  
        outRect.bottom = mItemOffset;  
    }  
}
```

You can call ItemDecoration like below code.

```
recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);  
  
GridLayoutManager lLayout = new GridLayoutManager(getActivity(), 2);  
  
ItemOffsetDecoration itemDecoration = new ItemOffsetDecoration(mActivity, R.dimen.item_offset);  
recyclerView.addItemDecoration(itemDecoration);  
  
recyclerView.setLayoutManager(lLayout);
```

and example item offset

```
<dimen name="item_offset">5dp</dimen>
```

```
<dimen name="item_offset">5dp</dimen>
```

第18章：RecyclerView 点击监听器

第18.1节：Kotlin 和 RxJava 示例

第一个示例用 Kotlin 重新实现，并使用 RxJava 以实现更简洁的交互。

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.support.v7.widget.RecyclerView
import rx.subjects.PublishSubject

public class SampleAdapter(private val items: Array<String>) : RecyclerView.Adapter<SampleAdapter.ViewHolder>() {

    // 从 rx.subjects 更改为不同的 subjects 以获得不同的行为// 例如 BehaviorSubject 允许在订阅时接收最后一个事件// 而 PublishSubject 只在订阅后发送事件，这对于点击事件是理想的

    public val itemClickStream: PublishSubject<View> = PublishSubject.create()

    override fun getItemCount(): Int {
        return items.size
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder? {
        val v = LayoutInflater.from(parent.getContext()).inflate(R.layout.text_row_item, parent, false);
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(items[position])
    }

    public inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        private val textView: TextView by lazy { view.findViewById(R.id.textView) as TextView }

        init {
            view.setOnClickListener { v -> itemClickStream.onNext(v) }
        }

        fun bind(text: String) {
            textView.text = text
        }
    }
}
```

使用非常简单。可以使用 RxJava 的功能在单独的线程上订阅。

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // 执行某些操作
    }
}
```

Chapter 18: RecyclerView onClickListeners

Section 18.1: Kotlin and RxJava example

First example reimplemented in Kotlin and using RxJava for cleaner interaction.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.support.v7.widget.RecyclerView
import rx.subjects.PublishSubject

public class SampleAdapter(private val items: Array<String>) : RecyclerView.Adapter<SampleAdapter.ViewHolder>() {

    // change to different subjects from rx.subjects to get different behavior
    // BehaviorSubject for example allows to receive last event on subscribe
    // PublishSubject sends events only after subscribing on the other hand which is desirable for
    // clicks

    public val itemClickStream: PublishSubject<View> = PublishSubject.create()

    override fun getItemCount(): Int {
        return items.size
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder? {
        val v = LayoutInflater.from(parent.getContext()).inflate(R.layout.text_row_item, parent, false);
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(items[position])
    }

    public inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        private val textView: TextView by lazy { view.findViewById(R.id.textView) as TextView }

        init {
            view.setOnClickListener { v -> itemClickStream.onNext(v) }
        }

        fun bind(text: String) {
            textView.text = text
        }
    }
}
```

Usage is quite simple then. It's possible to subscribe on separate thread using RxJava facilities.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

第18.2节：RecyclerView点击监听器

```
public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {  
  
    private GestureDetector gestureDetector;  
    private RecyclerTouchListener.ClickListener clickListener;  
  
    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final  
        RecyclerTouchListener.ClickListener clickListener) {  
        this.clickListener = clickListener;  
  
        gestureDetector = new GestureDetector(context, new  
            GestureDetector.SimpleOnGestureListener() {  
                @Override  
                public boolean onSingleTapUp(MotionEvent e) {  
                    return true;  
                }  
                @Override  
                public void onLongPress(MotionEvent e) {  
                    View child = recyclerView.findChildViewUnder(e.getX(), e.getY());  
                    if (child != null && clickListener != null) {  
                        clickListener.onLongClick(child, recyclerView.getChildPosition(child));  
                    }  
                }  
            });  
    }  
  
    @Override  
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {  
        View child = rv.findChildViewUnder(e.getX(), e.getY());  
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {  
            clickListener.onClick(child, rv.getChildPosition(child));  
        }  
        return false;  
    }  
  
    @Override  
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {  
    }  
  
    @Override  
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {  
    }  
  
    public interface ClickListener {  
        void onLongClick(View child, int childPosition);  
  
        void onClick(View child, int childPosition);  
    }  
}
```

在 MainActivity 中

```
RecyclerView recyclerView =(RecyclerView) findViewById(R.id.recyclerview);  
recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getActivity(),recyclerView, new  
RecyclerTouchListener.ClickListener() {  
    @Override  
    public void onLongClick(View child, int childPosition) {
```

Section 18.2: RecyclerView Click listener

```
public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {  
  
    private GestureDetector gestureDetector;  
    private RecyclerTouchListener.ClickListener clickListener;  
  
    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final  
        RecyclerTouchListener.ClickListener clickListener) {  
        this.clickListener = clickListener;  
  
        gestureDetector = new GestureDetector(context, new  
            GestureDetector.SimpleOnGestureListener() {  
                @Override  
                public boolean onSingleTapUp(MotionEvent e) {  
                    return true;  
                }  
                @Override  
                public void onLongPress(MotionEvent e) {  
                    View child = recyclerView.findChildViewUnder(e.getX(), e.getY());  
                    if (child != null && clickListener != null) {  
                        clickListener.onLongClick(child, recyclerView.getChildPosition(child));  
                    }  
                }  
            });  
    }  
  
    @Override  
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {  
        View child = rv.findChildViewUnder(e.getX(), e.getY());  
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {  
            clickListener.onClick(child, rv.getChildPosition(child));  
        }  
        return false;  
    }  
  
    @Override  
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {  
    }  
  
    @Override  
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {  
    }  
  
    public interface ClickListener {  
        void onLongClick(View child, int childPosition);  
  
        void onClick(View child, int childPosition);  
    }  
}
```

In MainActivity

```
RecyclerView recyclerView =(RecyclerView) findViewById(R.id.recyclerview);  
recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getActivity(),recyclerView, new  
RecyclerTouchListener.ClickListener() {  
    @Override  
    public void onLongClick(View child, int childPosition) {
```

```

        }

    @Override
    public void onClick(View child, int childPosition) {
        ...
    });

}

```

第18.3节：实现项目点击监听器的另一种方法

实现项目点击监听器的另一种方法是使用包含多个方法的接口，方法数量等于可点击视图的数量，并使用重写的点击监听器，如下所示。该方法更灵活，因为你可以为不同的视图设置点击监听器，并且可以相当容易地分别控制每个视图的点击逻辑。

```

public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.CustomHolder> {

    private ArrayList<Object> mObjects;
    private ClickInterface mClickInterface;

    public interface ClickInterface {
        void clickEventOne(Object obj);
        void clickEventTwo(Object obj1, Object obj2);
    }

    public void setClickInterface(ClickInterface clickInterface) {
        mClickInterface = clickInterface;
    }

    public CustomAdapter(){
        mList = new ArrayList<>();
    }

    public void addItems(ArrayList<Object> objects) {
        mObjects.clear();
        mObjects.addAll(objects);
        notifyDataSetChanged();
    }

    @Override
    public CustomHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, parent, false);
        return new CustomHolder(v);
    }

    @Override
    public void onBindViewHolder(CustomHolder holder, int position) {
        //make all even positions not clickable
        holder.firstClickListener.setClickable(position%2==0);
        holder.firstClickListener.setPosition(position);
        holder.secondClickListener.setPosition(position);
    }

    private class FirstClickListener implements View.OnClickListener {
        private int mPosition;
        private boolean mClickable;
    }
}

```

```

        }

    @Override
    public void onClick(View child, int childPosition) {
        ...
    });

}

```

Section 18.3: Another way to implement Item Click Listener

Another way to implement item click listener is to use interface with several methods, the number of which is equal to the number of clickable views, and use overridden click listeners as you can see below. This method is more flexible, because you can set click listeners to different views and quite easy control the click logic separately for each.

```

public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.CustomHolder> {

    private ArrayList<Object> mObjects;
    private ClickInterface mClickInterface;

    public interface ClickInterface {
        void clickEventOne(Object obj);
        void clickEventTwo(Object obj1, Object obj2);
    }

    public void setClickInterface(ClickInterface clickInterface) {
        mClickInterface = clickInterface;
    }

    public CustomAdapter(){
        mList = new ArrayList<>();
    }

    public void addItems(ArrayList<Object> objects) {
        mObjects.clear();
        mObjects.addAll(objects);
        notifyDataSetChanged();
    }

    @Override
    public CustomHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, parent, false);
        return new CustomHolder(v);
    }

    @Override
    public void onBindViewHolder(CustomHolder holder, int position) {
        //make all even positions not clickable
        holder.firstClickListener.setClickable(position%2==0);
        holder.firstClickListener.setPosition(position);
        holder.secondClickListener.setPosition(position);
    }

    private class FirstClickListener implements View.OnClickListener {
        private int mPosition;
        private boolean mClickable;
    }
}

```

```

void setPosition(int position) {
    mPosition = position;
}

void setClickable(boolean clickable) {
    mPosition = position;
}

@Override
public void onClick(View v) {
    if(mClickable) {
        mClickInterface.clickEventOne(mObjects.get(mPosition));
    }
}

private class SecondClickListener implements View.OnClickListener {
    private int mPosition;

    void setPosition(int position) {
        mPosition = position;
    }

    @Override
    public void onClick(View v) {
        mClickInterface.clickEventTwo(mObjects.get(mPosition), v);
    }
}

@Override
public int getItemCount() {
    return mObjects.size();
}

protected class CustomHolder extends RecyclerView.ViewHolder {
    FirstClickListener firstClickListener;
    SecondClickListener secondClickListener;
    View v1, v2;

    public DialogHolder(View itemView) {
        super(itemView);
        v1 = itemView.findViewById(R.id.v1);
        v2 = itemView.findViewById(R.id.v2);
        firstClickListener = new FirstClickListener();
        secondClickListener = new SecondClickListener();

        v1.setOnClickListener(firstClickListener);
        v2.setOnClickListener(secondClickListener);
    }
}
}

```

当你有一个适配器实例时，你可以设置点击监听器，用于监听每个视图的点击事件：

```

customAdapter.setClickInterface(new CustomAdapter.ClickInterface {
    @Override
    public void clickEventOne(Object obj) {
        // 你的实现代码
    }
    @Override
}

```

```

void setPosition(int position) {
    mPosition = position;
}

void setClickable(boolean clickable) {
    mPosition = position;
}

@Override
public void onClick(View v) {
    if(mClickable) {
        mClickInterface.clickEventOne(mObjects.get(mPosition));
    }
}

private class SecondClickListener implements View.OnClickListener {
    private int mPosition;

    void setPosition(int position) {
        mPosition = position;
    }

    @Override
    public void onClick(View v) {
        mClickInterface.clickEventTwo(mObjects.get(mPosition), v);
    }
}

@Override
public int getItemCount() {
    return mObjects.size();
}

protected class CustomHolder extends RecyclerView.ViewHolder {
    FirstClickListener firstClickListener;
    SecondClickListener secondClickListener;
    View v1, v2;

    public DialogHolder(View itemView) {
        super(itemView);
        v1 = itemView.findViewById(R.id.v1);
        v2 = itemView.findViewById(R.id.v2);
        firstClickListener = new FirstClickListener();
        secondClickListener = new SecondClickListener();

        v1.setOnClickListener(firstClickListener);
        v2.setOnClickListener(secondClickListener);
    }
}
}

```

And when you have an instance of adapter, you can set your click listener which listens to clicking on each of the views:

```

customAdapter.setClickInterface(new CustomAdapter.ClickInterface {
    @Override
    public void clickEventOne(Object obj) {
        // Your implementation here
    }
    @Override
}

```

```

public void clickEventTwo(Object obj1, Object obj2) {
    // 你的实现代码
}
);

```

第18.4节：新示例

```

public class SampleAdapter extends RecyclerView.Adapter<SampleAdapter.ViewHolder> {

    private String[] mDataSet;
    private OnRVIItemClickListener mListener;

    /**
     * 提供你所使用视图类型的引用（自定义ViewHolder）
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;

        public ViewHolder(View v) {
            super(v);
            // 为ViewHolder的视图定义点击监听器。
            v.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) { // 在这里处理点击事件
                    Log.d(TAG, "元素 " + getPosition() + " 被点击了。");
                    mListener.onRVIItemClicked(getPosition(), v); // 设置回调
                }
            });
            textView = (TextView) v.findViewById(R.id.textView);
        }

        public TextView getTextView() {
            return textView;
        }
    }

    /**
     * 初始化适配器的数据集。
     *
     * @param dataSet 包含用于填充RecyclerView视图的数据的字符串数组。
     */
    public SampleAdapter(String[] dataSet) {
        mDataSet = dataSet;
    }

    // 创建新视图（由布局管理器调用）
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // 创建一个新视图。
        View v = LayoutInflater.from(viewGroup.getContext())
            .inflate(R.layout.text_row_item, viewGroup, false);

        return new ViewHolder(v);
    }

    // 替换视图内容（由布局管理器调用）
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        // 从数据集中获取该位置的元素并替换视图内容
        // 使用该元素
        viewHolder.getTextView().setText(mDataSet[position]);
    }
}

```

```

public void clickEventTwo(Object obj1, Object obj2) {
    // Your implementation here
}
);

```

Section 18.4: New Example

```

public class SampleAdapter extends RecyclerView.Adapter<SampleAdapter.ViewHolder> {

    private String[] mDataSet;
    private OnRVIItemClickListener mListener;

    /**
     * Provide a reference to the type of views that you are using (custom ViewHolder)
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;

        public ViewHolder(View v) {
            super(v);
            // Define click listener for the ViewHolder's View.
            v.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) { // handle click events here
                    Log.d(TAG, "Element " + getPosition() + " clicked.");
                    mListener.onRVIItemClicked(getPosition(), v); // set callback
                }
            });
            textView = (TextView) v.findViewById(R.id.textView);
        }

        public TextView getTextView() {
            return textView;
        }
    }

    /**
     * Initialize the dataset of the Adapter.
     *
     * @param dataSet String[] containing the data to populate views to be used by RecyclerView.
     */
    public SampleAdapter(String[] dataSet) {
        mDataSet = dataSet;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Create a new view.
        View v = LayoutInflater.from(viewGroup.getContext())
            .inflate(R.layout.text_row_item, viewGroup, false);

        return new ViewHolder(v);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        // Get element from your dataset at this position and replace the contents of the view
        // with that element
        viewHolder.getTextView().setText(mDataSet[position]);
    }
}

```

```

}

// 返回数据集的大小 (由布局管理器调用)
@Override
public int getItemCount() {
    return mDataSet.length;
}

public void setOnRVClickListener(OnRVIItemClickListener) {
    mListener = OnRVIItemClickListener;
}

public interface OnRVIItemClickListener {
    void onRVIItemClicked(int position, View v);
}

```

第18.5节：简单的OnLongClick和OnClick示例

首先，实现你的视图持有者：

实现 View.OnClickListener, View.OnLongClickListener

然后，按如下方式注册监听器：

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

接下来，重写监听器如下：

```

@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}

```

最后，添加以下代码：

```

public void setOnItemClickListener(onClickListner onclicklistner) {
    SampleAdapter.onclicklistner = onclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}

public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}

```

适配器演示

包 adaptor;

```

}

// Return the size of your dataset (invoked by the layout manager)
@Override
public int getItemCount() {
    return mDataSet.length;
}

public void setOnRVClickListener(OnRVIItemClickListener) {
    mListener = OnRVIItemClickListener;
}

public interface OnRVIItemClickListener {
    void onRVIItemClicked(int position, View v);
}

```

Section 18.5: Easy OnLongClick and OnClickListener Example

First of all, implement your view holder:

implements View.OnClickListener, View.OnLongClickListener

Then, register the listeners as follows:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Next, override the listeners as follows:

```

@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}

```

And finally, add the following code:

```

public void setOnItemClickListener(onClickListner onclicklistner) {
    SampleAdapter.onclicklistner = onclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}

public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}

```

Adaptor demo

package adaptor;

```

导入 android.annotation.SuppressLint;
导入 android.content.Context;
导入 android.support.v7.widget.RecyclerView;
导入 android.view.LayoutInflater;
导入 android.view.View;
导入 android.view.ViewGroup;
导入 android.widget.TextView;

导入 com.wings.example.recyclerview.MainActivity;
导入 com.wings.example.recyclerview.R;

导入 java.util.ArrayList;

公共类 SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    上下文 context;
    私有 ArrayList<String> arrayList;
    私有静态 onClickListner onclicklistner;
    私有静态常量 int VIEW_HEADER = 0;
    私有静态常量 int VIEW_NORMAL = 1;
    私有 View headerView;

    公共 SampleAdapter(Context context) {
        this.context = context;
        arrayList = MainActivity.arrayList;
    }

    public class HeaderViewHolder extends RecyclerView.ViewHolder {
        public HeaderViewHolder(View itemView) {
            super(itemView);
        }
    }

    public class ItemViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,
    View.OnLongClickListener {
        TextView txt_pos;
        SampleAdapter sampleAdapter;

        public ItemViewHolder(View itemView, SampleAdapter sampleAdapter) {
            super(itemView);

            itemView.setOnClickListener(this);
            itemView.setOnLongClickListener(this);

            txt_pos = (TextView) itemView.findViewById(R.id.txt_pos);
            this.sampleAdapter = sampleAdapter;

            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            onclicklistner.onItemClick(getAdapterPosition(), v);
        }

        @Override
        public boolean onLongClick(View v) {
            onclicklistner.onItemLongClick(getAdapterPosition(), v);
            return true;
        }
    }

    public void setOnItemClickListener(OnClickListner onclicklistner) {

```

```

import android.annotation.SuppressLint;
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.wings.example.recyclerview.MainActivity;
import com.wings.example.recyclerview.R;

import java.util.ArrayList;

public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    Context context;
    private ArrayList<String> arrayList;
    private static onClickListner onclicklistner;
    private static final int VIEW_HEADER = 0;
    private static final int VIEW_NORMAL = 1;
    private View headerView;

    public SampleAdapter(Context context) {
        this.context = context;
        arrayList = MainActivity.arrayList;
    }

    public class HeaderViewHolder extends RecyclerView.ViewHolder {
        public HeaderViewHolder(View itemView) {
            super(itemView);
        }
    }

    public class ItemViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,
    View.OnLongClickListener {
        TextView txt_pos;
        SampleAdapter sampleAdapter;

        public ItemViewHolder(View itemView, SampleAdapter sampleAdapter) {
            super(itemView);

            itemView.setOnClickListener(this);
            itemView.setOnLongClickListener(this);

            txt_pos = (TextView) itemView.findViewById(R.id.txt_pos);
            this.sampleAdapter = sampleAdapter;

            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            onclicklistner.onItemClick(getAdapterPosition(), v);
        }

        @Override
        public boolean onLongClick(View v) {
            onclicklistner.onItemLongClick(getAdapterPosition(), v);
            return true;
        }
    }

    public void setOnItemClickListener(OnClickListner onclicklistner) {

```

```

SampleAdapter.onclicklistner = onclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}

public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}

@Override
public int getItemCount() {
    return arrayList.size()+1;
}

@Override
public int getItemViewType(int position) {
    return position == 0 ? VIEW_HEADER : VIEW_NORMAL;
}

@SuppressLint("InflateParams")
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
    if (viewType == VIEW_HEADER) {
        return new HeaderViewHolder(headerView);
    } 否则 {
        View view =
LayoutInflator.from(viewGroup.getContext()).inflate(R.layout.custom_recycler_row_sample_item,
viewGroup, false);
        return new ItemViewHolder(view, this);
    }
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    if (viewHolder.getItemViewType() == VIEW_HEADER) {
        return;
    } 否则 {
ItemViewHolder itemViewHolder = (ItemViewHolder) viewHolder;
        itemViewHolder.txt_pos.setText(arrayList.get(position-1));
    }
}

```

上面的示例代码可以通过以下代码调用：

```

sampleAdapter.setOnItemClickListener(new SampleAdapter.onClickListner() {
    @Override
    public void onItemClick(int position, View v) {
        position = position+1;//因为我们添加了头部
        Log.e(TAG + "ON ITEM CLICK", position + "");
        Snackbar.make(v, "点击项 "+position, Snackbar.LENGTH_LONG).show();
    }

    @Override
    public void onItemLongClick(int position, View v) {
        position = position+1;//因为我们添加了头部
        Log.e(TAG + "ON ITEM LONG CLICK", position + "");
        Snackbar.make(v, "长按项 "+position, Snackbar.LENGTH_LONG).show();
    }
})

```

```

SampleAdapter.onclicklistner = onclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}

public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}

@Override
public int getItemCount() {
    return arrayList.size()+1;
}

@Override
public int getItemViewType(int position) {
    return position == 0 ? VIEW_HEADER : VIEW_NORMAL;
}

@SuppressLint("InflateParams")
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
    if (viewType == VIEW_HEADER) {
        return new HeaderViewHolder(headerView);
    } else {
        View view =
LayoutInflator.from(viewGroup.getContext()).inflate(R.layout.custom_recycler_row_sample_item,
viewGroup, false);
        return new ItemViewHolder(view, this);
    }
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    if (viewHolder.getItemViewType() == VIEW_HEADER) {
        return;
    } else {
ItemViewHolder itemViewHolder = (ItemViewHolder) viewHolder;
        itemViewHolder.txt_pos.setText(arrayList.get(position-1));
    }
}

```

The example code above can be called by the following code:

```

sampleAdapter.setOnItemClickListener(new SampleAdapter.onClickListner() {
    @Override
    public void onItemClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM CLICK", position + "");
        Snackbar.make(v, "On item click "+position, Snackbar.LENGTH_LONG).show();
    }

    @Override
    public void onItemLongClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM LONG CLICK", position + "");
        Snackbar.make(v, "On item longclick "+position, Snackbar.LENGTH_LONG).show();
    }
})

```

```
    }  
});
```

第18.6节：项点击监听器

要实现项点击监听器和/或项长按监听器，可以在适配器中创建一个接口：

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {  
  
    public interface OnItemClickListener {  
  
        void onItemClick(int position, View view, CustomObject object);  
    }  
  
    public interface OnItemLongClickListener {  
  
        boolean onItemLongClick(int position, View view, CustomObject object);  
    }  
  
    public final class ViewHolder extends RecyclerView.ViewHolder {  
  
        public ViewHolder(View itemView) {  
            super(itemView);  
            final int position = getAdapterPosition();  
  
            itemView.setOnClickListener(new View.OnClickListener() {  
                @Override  
                public void onClick(View view) {  
                    if(mOnItemClickListener != null) {  
                        mOnItemClickListener.onItemClick(position, view, mDataSet.get(position));  
                    }  
                }  
            });  
  
            itemView.setOnLongClickListener(new View.OnLongClickListener() {  
                @Override  
                public boolean onLongClick(View view) {  
                    if(mOnItemLongClickListener != null) {  
                        return mOnItemLongClickListener.onItemLongClick(position, view,  
                            mDataSet.get(position));  
                    }  
                }  
            });  
        }  
  
        private List<CustomObject> mDataSet;  
  
        private OnItemClickListener mOnItemClickListener;  
        private OnItemLongClickListener mOnItemLongClickListener;  
  
        public CustomAdapter(List<CustomObject> dataSet) {  
            mDataSet = dataSet;  
        }  
  
        @Override  
        public CustomAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
            View view = LayoutInflater.from(parent.getContext())  
                .inflate(R.layout.view_item_custom, parent, false);  
            return new ViewHolder(view);  
        }
```

```
    }  
});
```

Section 18.6: Item Click Listeners

To implement an item click listener and/or an item long click listener, you can create an interface in your adapter:

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {  
  
    public interface OnItemClickListener {  
  
        void onItemClick(int position, View view, CustomObject object);  
    }  
  
    public interface OnItemLongClickListener {  
  
        boolean onItemLongClick(int position, View view, CustomObject object);  
    }  
  
    public final class ViewHolder extends RecyclerView.ViewHolder {  
  
        public ViewHolder(View itemView) {  
            super(itemView);  
            final int position = getAdapterPosition();  
  
            itemView.setOnClickListener(new View.OnClickListener() {  
                @Override  
                public void onClick(View view) {  
                    if(mOnItemClickListener != null) {  
                        mOnItemClickListener.onItemClick(position, view, mDataSet.get(position));  
                    }  
                }  
            });  
  
            itemView.setOnLongClickListener(new View.OnLongClickListener() {  
                @Override  
                public boolean onLongClick(View view) {  
                    if(mOnItemLongClickListener != null) {  
                        return mOnItemLongClickListener.onItemLongClick(position, view,  
                            mDataSet.get(position));  
                    }  
                }  
            });  
        }  
  
        private List<CustomObject> mDataSet;  
  
        private OnItemClickListener mOnItemClickListener;  
        private OnItemLongClickListener mOnItemLongClickListener;  
  
        public CustomAdapter(List<CustomObject> dataSet) {  
            mDataSet = dataSet;  
        }  
  
        @Override  
        public CustomAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
            View view = LayoutInflater.from(parent.getContext())  
                .inflate(R.layout.view_item_custom, parent, false);  
            return new ViewHolder(view);  
        }
```

```

}

@Override
public void onBindViewHolder(CustomAdapter.ViewHolder holder, int position) {
    // 绑定视图
}

@Override
public int getItemCount() {
    return mDataSet.size();
}

public void setOnItemClickListener(OnItemClickListener listener) {
    mOnItemClickListener = listener;
}

public void setOnItemLongClickListener(OnItemLongClickListener listener) {
    mOnItemLongClickListener = listener;
}

}

```

然后你可以在创建适配器实例后设置点击监听器：

```

customAdapter.setOnItemClickListener(new CustomAdapter.OnItemClickListener {
    @Override
    public void onItemClick(int position, View view, CustomObject object) {
        // 你的实现代码
    }
});

customAdapter.setOnItemLongClickListener(new CustomAdapter.OnItemLongClickListener {
    @Override
    public boolean onItemSelected(int position, View view, CustomObject object) {
        // 你的实现代码
        return true;
    }
});

```

```

}

@Override
public void onBindViewHolder(CustomAdapter.ViewHolder holder, int position) {
    // Bind views
}

@Override
public int getItemCount() {
    return mDataSet.size();
}

public void setOnItemClickListener(OnItemClickListener listener) {
    mOnItemClickListener = listener;
}

public void setOnItemLongClickListener(OnItemLongClickListener listener) {
    mOnItemLongClickListener = listener;
}

}

```

Then you can set your click listeners after you create an instance of the adapter:

```

customAdapter.setOnItemClickListener(new CustomAdapter.OnItemClickListener {
    @Override
    public void onItemClick(int position, View view, CustomObject object) {
        // Your implementation here
    }
});

customAdapter.setOnItemLongClickListener(new CustomAdapter.OnItemLongClickListener {
    @Override
    public boolean onItemSelected(int position, View view, CustomObject object) {
        // Your implementation here
        return true;
    }
});

```

第19章：RecyclerView和布局管理器

第19.1节：为带有网格布局管理器的RecyclerView添加头部视图

要为带有网格布局的RecyclerView添加头部，首先需要告诉适配器头部视图是第一个位置——而不是用于内容的标准单元格。接下来，布局管理器必须被告知第一个位置的跨度应等于整个列表的*跨度计数。^{*}

以一个普通的RecyclerView.Adapter类为例，按如下方式配置：

```
public class HeaderAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int ITEM_VIEW_TYPE_HEADER = 0;
    private static final int ITEM_VIEW_TYPE_ITEM = 1;

    private List<YourModel> mModelList;

    public HeaderAdapter (List<YourModel> modelList) {
        mModelList = modelList;
    }

    public boolean isHeader(int position) {
        return position == 0;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());

        if (viewType == ITEM_VIEW_TYPE_HEADER) {
            View headerView = inflater.inflate(R.layout.header, parent, false);
            return new HeaderHolder(headerView);
        }

        View cellView = inflater.inflate(R.layout.gridcell, parent, false);
        return new ModelHolder(cellView);
    }

    @Override
    public int getItemViewType(int position) {
        return isHeader(position) ? ITEM_VIEW_TYPE_HEADER : ITEM_VIEW_TYPE_ITEM;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder h, int position) {
        if (isHeader(position)) {
            return;
        }

        final YourModel 模型 = mModelList.get(position -1); // 减去1以适应头部

        ModelHolder holder = (ModelHolder) h;
        // 像往常一样用你的模型数据填充holder
    }

    @Override
```

Chapter 19: RecyclerView and LayoutManagers

Section 19.1: Adding header view to recyclerview with gridlayout manager

To add a header to a recyclerview with a gridlayout, first the adapter needs to be told that the header view is the first position - rather than the standard cell used for the content. Next, the layout manager must be told that the first position should have a span equal to the *span count of the entire list. *

Take a regular RecyclerView.Adapter class and configure it as follows:

```
public class HeaderAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int ITEM_VIEW_TYPE_HEADER = 0;
    private static final int ITEM_VIEW_TYPE_ITEM = 1;

    private List<YourModel> mModelList;

    public HeaderAdapter (List<YourModel> modelList) {
        mModelList = modelList;
    }

    public boolean isHeader(int position) {
        return position == 0;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());

        if (viewType == ITEM_VIEW_TYPE_HEADER) {
            View headerView = inflater.inflate(R.layout.header, parent, false);
            return new HeaderHolder(headerView);
        }

        View cellView = inflater.inflate(R.layout.gridcell, parent, false);
        return new ModelHolder(cellView);
    }

    @Override
    public int getItemViewType(int position) {
        return isHeader(position) ? ITEM_VIEW_TYPE_HEADER : ITEM_VIEW_TYPE_ITEM;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder h, int position) {
        if (isHeader(position)) {
            return;
        }

        final YourModel model = mModelList.get(position -1 ); // Subtract 1 for header

        ModelHolder holder = (ModelHolder) h;
        // populate your holder with data from your model as usual
    }

    @Override
```

```

public int getItemCount() {
    return _categories.size() + 1; // 为头部添加一个
}

```

然后在activity/fragment中：

```

final HeaderAdapter adapter = new HeaderAdapter (mModelList);
final GridLayoutManager manager = new GridLayoutManager();
manager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
    @Override
    public int getSpanSize(int position) {
        return adapter.isHeader(position) ? manager.getSpanCount() : 1;
    }
});
mRecyclerView.setLayoutManager(manager);
mRecyclerView.setAdapter(adapter);

```

同样的方法也可以用来添加页脚，作为头部的补充或替代。

来源：[Chiu-Ki Chan的Square Island博客](#)

第19.2节：具有动态跨度计数的GridLayoutManager

在使用 GridLayout 布局管理器创建 RecyclerView 时，必须在构造函数中指定跨度数。跨度数指的是列数。这种方式相当笨重，且没有考虑到更大屏幕尺寸或屏幕方向。一个方法是为不同屏幕尺寸创建多个布局。另一种更动态的方法如下所示。

首先，我们创建一个自定义的 RecyclerView 类，如下所示：

```

public class AutofitRecyclerView extends RecyclerView {
    private GridLayoutManager manager;
    private int columnWidth = -1;

    public AutofitRecyclerView(Context context) {
        super(context);
        init(context, null);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context, attrs);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context, attrs);
    }

    private void init(Context context, AttributeSet attrs) {
        if (attrs != null) {
            int[] attrsArray = {
                android.R.attr.columnWidth
            };
            TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
            columnWidth = array.getDimensionPixelSize(0, -1);
            array.recycle();
        }
    }
}

```

```

public int getItemCount() {
    return _categories.size() + 1; // add one for the header
}

```

Then in the activity/fragment:

```

final HeaderAdapter adapter = new HeaderAdapter (mModelList);
final GridLayoutManager manager = new GridLayoutManager();
manager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
    @Override
    public int getSpanSize(int position) {
        return adapter.isHeader(position) ? manager.getSpanCount() : 1;
    }
});
mRecyclerView.setLayoutManager(manager);
mRecyclerView.setAdapter(adapter);

```

The same approach can be used add a footer in addition to or instead of a header.

Source: [Chiu-Ki Chan's Square Island blog](#)

Section 19.2: GridLayoutManager with dynamic span count

When creating a.recyclerview with a gridlayout layout manager you have to specify the span count in the constructor. Span count refers to the number of columns. This is fairly clunky and doesn't take into account larger screen sizes or screen orientation. One approach is to create multiple layouts for the various screen sizes. Another more dynamic approach can be seen below.

First we create a custom RecyclerView class as follows:

```

public class AutofitRecyclerView extends RecyclerView {
    private GridLayoutManager manager;
    private int columnWidth = -1;

    public AutofitRecyclerView(Context context) {
        super(context);
        init(context, null);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context, attrs);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context, attrs);
    }

    private void init(Context context, AttributeSet attrs) {
        if (attrs != null) {
            int[] attrsArray = {
                android.R.attr.columnWidth
            };
            TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
            columnWidth = array.getDimensionPixelSize(0, -1);
            array.recycle();
        }
    }
}

```

```

manager = new GridLayoutManager(getContext(), 1);
setLayoutManager(manager);

}

@Override
protected void onMeasure(int widthSpec, int heightSpec) {
    super.onMeasure(widthSpec, heightSpec);
    if (columnWidth > 0) {
        int spanCount = Math.max(1, getMeasuredWidth() / columnWidth);
        manager.setSpanCount(spanCount);
    }
}
}

```

此类用于确定RecyclerView中可以容纳多少列。要使用它，您需要将其放入您的layout.xml中，格式如下：

```

<?xml version="1.0" encoding="utf-8"?>
<com.path.to.your.class.autofitRecyclerView.AutofitRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/auto_fit_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="200dp"
    android:clipToPadding="false"
/>

```

注意我们使用了 columnWidth 属性。RecyclerView 需要它来确定有多少列可以适应可用空间。

在你的 activity/fragment 中，只需获取 RecyclerView 的引用并为其设置适配器（以及你想添加的任何项目装饰或动画）。不要设置布局管理器

```

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());

```

(其中 MyAdapter 是你的适配器类)

现在你有了一个 RecyclerView，它会根据屏幕大小自动调整跨度数（即列数）。最后，你可能想要将 RecyclerView 中的列居中（默认情况下它们是对齐到 layout_start）。你可以通过稍微修改 AutofitRecyclerView 类来实现。首先在 RecyclerView 中创建一个内部类。这个类将继承自 GridLayoutManager。它会在左右两边添加足够的内边距以居中行：

```

public class AutofitRecyclerView extends RecyclerView {

    // 等等，见上文

    private class CenteredGridLayoutManager extends GridLayoutManager {

        public CenteredGridLayoutManager(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
            super(context, attrs, defStyleAttr, defStyleRes);
        }

        public CenteredGridLayoutManager(Context context, int spanCount) {
            super(context, spanCount);
        }
    }
}

```

```

manager = new GridLayoutManager(getContext(), 1);
setLayoutManager(manager);

}

@Override
protected void onMeasure(int widthSpec, int heightSpec) {
    super.onMeasure(widthSpec, heightSpec);
    if (columnWidth > 0) {
        int spanCount = Math.max(1, getMeasuredWidth() / columnWidth);
        manager.setSpanCount(spanCount);
    }
}
}

```

This class determines how many columns can fit into the recyclerview. To use it you will need to put it into your layout.xml as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<com.path.to.your.class.autofitRecyclerView.AutofitRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/auto_fit_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="200dp"
    android:clipToPadding="false"
/>

```

Notice that we use the columnWidth attribute. The recyclerview will need it to determine how many columns will fit into the available space.

In your activity/fragment you just get a reference to the recyclerview and set an adapter to it (and any item decorations or animations that you want to add). **DO NOT SET A LAYOUT MANAGER**

```

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());

```

(where MyAdapter is your adapter class)

You now have a recyclerview that will adjust the spancount (ie columns) to fit the screen size. As a final addition you might want to center the columns in the recyclerview (by default they are aligned to layout_start). You can do that by modifying the AutofitRecyclerView class a little. Start by creating an inner class in the recyclerview. This will be a class that extends from GridLayoutManager. It will add enough padding to the left and right in order to center the rows:

```

public class AutofitRecyclerView extends RecyclerView {

    // etc see above

    private class CenteredGridLayoutManager extends GridLayoutManager {

        public CenteredGridLayoutManager(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
            super(context, attrs, defStyleAttr, defStyleRes);
        }

        public CenteredGridLayoutManager(Context context, int spanCount) {
            super(context, spanCount);
        }
    }
}

```

```

public CenteredGridLayoutManager(Context context, int spanCount, int orientation, boolean
reverseLayout) {
    super(context, spanCount, orientation, reverseLayout);
}

@Override
public int getPaddingLeft() {
    final int totalItemWidth = columnWidth * getSpanCount();
    if (totalItemWidth >= AutofitRecyclerView.this.getMeasuredWidth()) {
        return super.getPaddingLeft(); // 不做任何处理
    } 否则 {
        return Math.round((AutofitRecyclerView.this.getMeasuredWidth() / (1f +
getSpanCount())) - (totalItemWidth / (1f + getSpanCount())));
    }
}

@Override
public int getPaddingRight() {
    return getPaddingLeft();
}
}
}

```

然后当你在 AutofitRecyclerView 中设置 LayoutManager 时，使用 CenteredGridLayoutManager，如下所示：

```

private void init(Context context, AttributeSet attrs) {
    if (attrs != null) {
        int[] attrsArray = {
            android.R.attr.columnWidth
        };
        TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
        columnWidth = array.getDimensionPixelSize(0, -1);
        array.recycle();
    }

    manager = new CenteredGridLayoutManager(getContext(), 1);
    setLayoutManager(manager);
}

```

就是这样！你已经拥有了一个动态列数、居中对齐的基于GridLayoutManager的RecyclerView。

来源：

- [陈秋祺的Square Island博客](#)
- [StackOverflow](#)

第19.3节：使用LinearLayoutManager的简单列表

此示例通过使用自定义Place对象的ArrayList作为数据集，添加了带有图片和名称的地点列表。

活动布局

活动/片段的布局或RecyclerView使用的位置只需包含RecyclerView。无需ScrollView或特定布局。

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```

public CenteredGridLayoutManager(Context context, int spanCount, int orientation, boolean
reverseLayout) {
    super(context, spanCount, orientation, reverseLayout);
}

@Override
public int getPaddingLeft() {
    final int totalItemWidth = columnWidth * getSpanCount();
    if (totalItemWidth >= AutofitRecyclerView.this.getMeasuredWidth()) {
        return super.getPaddingLeft(); // do nothing
    } else {
        return Math.round((AutofitRecyclerView.this.getMeasuredWidth() / (1f +
getSpanCount())) - (totalItemWidth / (1f + getSpanCount())));
    }
}

@Override
public int getPaddingRight() {
    return getPaddingLeft();
}
}

```

Then when you set the LayoutManager in the AutofitRecyclerView use the CenteredGridLayoutManager as follows:

```

private void init(Context context, AttributeSet attrs) {
    if (attrs != null) {
        int[] attrsArray = {
            android.R.attr.columnWidth
        };
        TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
        columnWidth = array.getDimensionPixelSize(0, -1);
        array.recycle();
    }

    manager = new CenteredGridLayoutManager(getContext(), 1);
    setLayoutManager(manager);
}

```

And that's it! You have a dynamic spancount, center aligned gridlayoutmanager based recyclerview.

Sources:

- [Chiu-Ki Chan's Square Island blog](#)
- [StackOverflow](#)

Section 19.3: Simple list with LinearLayoutManager

This example adds a list of places with image and name by using an `ArrayList` of custom Place objects as dataset.

Activity layout

The layout of the activity / fragment or where the RecyclerView is used only has to contain the RecyclerView. There is no ScrollView or a specific layout needed.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/my_recycler_view"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />  
  
</RelativeLayout>
```

定义数据模型

你可以使用任何类或基本数据类型作为模型，比如int、String、float[]或CustomObject。RecyclerView将引用这些对象/基本类型的List。

当列表项包含不同类型，如文本、数字、图片（如本例中的地点）时，通常使用自定义对象是个好主意。

```
public class Place {  
    // 这些字段将在列表项中显示  
    private Bitmap image;  
    private String name;  
  
    // 典型构造函数  
    public Place(Bitmap image, String name) {  
        this.image = image;  
        this.name = name;  
    }  
  
    // 获取器  
    public Bitmap getImage() {  
        return image;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

列表项布局

你必须指定一个用于每个列表项的 XML 布局文件。在此示例中，ImageView 用于显示图像，TextView 用于显示名称。LinearLayout 将 ImageView 放置在左侧，TextView 放置在图像的右侧。

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:gravity="center_vertical"  
    android:orientation="horizontal"  
    android:padding="8dp">  
  
    <ImageView  
        android:id="@+id/image"  
        android:layout_width="36dp"  
        android:layout_height="36dp"  
        android:layout_marginEnd="8dp"  
        android:layout_marginRight="8dp" />  
  
    <TextView  
        android:id="@+id/name"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content" />
```

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/my_recycler_view"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />  
  
</RelativeLayout>
```

Define the data model

You could use any class or primitive data type as a model, like int, String, float[] or CustomObject. The RecyclerView will refer to a List of this objects / primitives.

When a list item refers to different data types like text, numbers, images (as in this example with places), it is often a good idea to use a custom object.

```
public class Place {  
    // these fields will be shown in a list item  
    private Bitmap image;  
    private String name;  
  
    // typical constructor  
    public Place(Bitmap image, String name) {  
        this.image = image;  
        this.name = name;  
    }  
  
    // getters  
    public Bitmap getImage() {  
        return image;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

List item layout

You have to specify a xml layout file that will be used for each list item. In this example, an ImageView is used for the image and a TextView for the name. The LinearLayout positions the ImageView at the left and the TextView right to the image.

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:gravity="center_vertical"  
    android:orientation="horizontal"  
    android:padding="8dp">  
  
    <ImageView  
        android:id="@+id/image"  
        android:layout_width="36dp"  
        android:layout_height="36dp"  
        android:layout_marginEnd="8dp"  
        android:layout_marginRight="8dp" />  
  
    <TextView  
        android:id="@+id/name"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

创建 RecyclerView 适配器和 ViewHolder

接下来，你需要继承 RecyclerView.Adapter 和 RecyclerView.ViewHolder。一个常见的类结构是：

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    // ...  
  
    public class ViewHolder extends RecyclerView.ViewHolder {  
        // ...  
    }  
}
```

首先，我们实现 ViewHolder。它仅继承默认构造函数，并将所需的视图保存到一些字段中：

```
public class ViewHolder extends RecyclerView.ViewHolder {  
    private ImageView imageView;  
    private TextView nameView;  
  
    public ViewHolder(View itemView) {  
        super(itemView);  
  
        imageView = (ImageView) itemView.findViewById(R.id.image);  
        nameView = (TextView) itemView.findViewById(R.id.name);  
    }  
}
```

适配器的构造函数设置所使用的数据集：

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    private List<Place> mPlaces;  
  
    public PlaceListAdapter(List<Place> contacts) {  
        mPlaces = contacts;  
    }  
  
    // ...  
}
```

为了使用我们自定义的列表项布局，我们重写了方法onCreateViewHolder(...). 在本例中，布局文件名为place_list_item.xml。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    // ...  
  
    @Override  
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        View view = LayoutInflater.from(parent.getContext()).inflate(  
            R.layout.place_list_item,  
            parent,  
            false  
        );  
        return new ViewHolder(view);  
    }  
  
    // ...  
}
```

```
</LinearLayout>
```

Create a RecyclerView adapter and ViewHolder

Next, you have to inherit the RecyclerView.Adapter and the RecyclerView.ViewHolder. A usual class structure would be:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    // ...  
  
    public class ViewHolder extends RecyclerView.ViewHolder {  
        // ...  
    }  
}
```

First, we implement the ViewHolder. It only inherits the default constructor and saves the needed views into some fields:

```
public class ViewHolder extends RecyclerView.ViewHolder {  
    private ImageView imageView;  
    private TextView nameView;  
  
    public ViewHolder(View itemView) {  
        super(itemView);  
  
        imageView = (ImageView) itemView.findViewById(R.id.image);  
        nameView = (TextView) itemView.findViewById(R.id.name);  
    }  
}
```

The adapter's constructor sets the used dataset:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    private List<Place> mPlaces;  
  
    public PlaceListAdapter(List<Place> contacts) {  
        mPlaces = contacts;  
    }  
  
    // ...  
}
```

To use our custom list item layout, we override the method onCreateViewHolder(...). In this example, the layout file is called place_list_item.xml.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {  
    // ...  
  
    @Override  
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        View view = LayoutInflater.from(parent.getContext()).inflate(  
            R.layout.place_list_item,  
            parent,  
            false  
        );  
        return new ViewHolder(view);  
    }  
  
    // ...  
}
```

```
}
```

在 `onBindViewHolder(...)` 中，我们实际上设置视图的内容。我们通过在 `List` 中找到给定位置的模型，然后在 `ViewHolder` 的视图上设置图片和名称。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public void onBindViewHolder(PlaceListAdapter.ViewHolder viewHolder, int position) {
        Place place = mPlaces.get(position);

        viewHolder.nameView.setText(place.getName());
        viewHolder.imageView.setImageBitmap(place.getImage());
    }

    // ...
}
```

我们还需要实现 `getItemCount()`，它简单地返回 `List` 的大小。

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(生成随机数据)

在此示例中，我们将生成一些随机地点。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    List<Place> places = randomPlaces(5);

    // ...

    private List<Place> randomPlaces(int amount) {
        List<Place> places = new ArrayList<>();
        for (int i = 0; i < amount; i++) {
            places.add(new Place(
                BitmapFactory.decodeResource(getResources(), Math.random() > 0.5 ?
                    R.drawable.ic_account_grey600_36dp :
                    R.drawable.ic_android_grey600_36dp
            ),
            "地点 #" + (int) (Math.random() * 1000)
        );
    }
    return places;
}
```

将 `RecyclerView` 与 `PlaceListAdapter` 及数据集连接

```
}
```

In the `onBindViewHolder(...)`, we actually set the views' contents. We get the used model by finding it in the `List` at the given position and then set image and name on the `ViewHolder`'s views.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public void onBindViewHolder(PlaceListAdapter.ViewHolder viewHolder, int position) {
        Place place = mPlaces.get(position);

        viewHolder.nameView.setText(place.getName());
        viewHolder.imageView.setImageBitmap(place.getImage());
    }

    // ...
}
```

We also need to implement `getItemCount()`, which simply return the `List`'s size.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Generate random data)

For this example, we'll generate some random places.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    List<Place> places = randomPlaces(5);

    // ...

    private List<Place> randomPlaces(int amount) {
        List<Place> places = new ArrayList<>();
        for (int i = 0; i < amount; i++) {
            places.add(new Place(
                BitmapFactory.decodeResource(getResources(), Math.random() > 0.5 ?
                    R.drawable.ic_account_grey600_36dp :
                    R.drawable.ic_android_grey600_36dp
            ),
            "Place #" + (int) (Math.random() * 1000)
        ));
    }
    return places;
}
```

Connect the `RecyclerView` with the `PlaceListAdapter` and the dataset

将RecyclerView与适配器连接非常简单。你需要设置LinearLayoutManager作为布局管理器以实现列表布局。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    // ...  
  
    RecyclerView recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);  
    recyclerView.setAdapter(new PlaceListAdapter(places));  
    recyclerView.setLayoutManager(new LinearLayoutManager(this));  
}
```

完成！

第19.4节：StaggeredGridLayoutManager（错落网格布局管理器）

1. 在布局xml文件中创建你的RecyclerView：

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/recycleView"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

2. 创建用于保存数据的模型类：

```
public class PintrestItem {  
    String url;  
    public PintrestItem(String url, String name){  
        this.url=url;  
        this.name=name;  
    }  
    public String getUrl(){  
        return url;  
    }  
  
    public String getName(){  
        return name;  
    }  
    String name;  
}
```

3. 创建一个布局文件以容纳 RecyclerView 项目：

```
<ImageView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:adjustViewBounds="true"  
    android:scaleType="centerCrop"  
    android:id="@+id/imageView"/>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:gravity="center"  
    android:id="@+id/name"  
    android:layout_gravity="center"  
    android:textColor="@android:color/white"/>
```

4. 为RecyclerView创建适配器类：

Connecting a RecyclerView with an adapter is very easy. You have to set the LinearLayoutManager as layout manager to achieve the list layout.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    // ...  
  
    RecyclerView recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);  
    recyclerView.setAdapter(new PlaceListAdapter(places));  
    recyclerView.setLayoutManager(new LinearLayoutManager(this));  
}  
  
Done!
```

Section 19.4: StaggeredGridLayoutManager

1. Create your RecyclerView in your layout xml file:

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/recycleView"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

2. Create your Model class for holding your data:

```
public class PintrestItem {  
    String url;  
    public PintrestItem(String url, String name){  
        this.url=url;  
        this.name=name;  
    }  
    public String getUrl(){  
        return url;  
    }  
  
    public String getName(){  
        return name;  
    }  
    String name;  
}
```

3. Create a layout file to hold RecyclerView items:

```
<ImageView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:adjustViewBounds="true"  
    android:scaleType="centerCrop"  
    android:id="@+id/imageView"/>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:gravity="center"  
    android:id="@+id/name"  
    android:layout_gravity="center"  
    android:textColor="@android:color/white"/>
```

4. Create the adapter class for the RecyclerView:

```

public class PintrestAdapter extends
RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem>images;
Picasso picasso;
Context context;
public PintrestAdapter(ArrayList<PintrestItem>images,Context context){
    this.images=images;
    picasso=Picasso.with(context);
    this.context=context;
}

@Override
public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_layout_item,parent,false);
    return new PintrestViewHolder(view);
}

@Override
public void onBindViewHolder(PintrestViewHolder holder, int position) {
    picasso.load(images.get(position).getUrl()).into(holder.imageView);
    holder.tv.setText(images.get(position).getName());
}

@Override
public int getItemCount() {
    return images.size();
}

public class PintrestViewHolder extends RecyclerView.ViewHolder{
    ImageView imageView;
    TextView tv;
    public PintrestViewHolder(View itemView) {
        super(itemView);
    }
    imageView=(ImageView)itemView.findViewById(R.id.imageView);
    tv=(TextView)itemView.findViewById(R.id.name);
}
}

```

5. 在您的活动或片段中实例化 RecyclerView :

```

RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recyclerView);
// 创建 StaggeredGridLayoutManager 实例, 设置为 2 行, 即跨度数, 并提供方向
StaggeredGridLayoutManager layoutManager=new new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
// 创建虚拟数据并添加到您的 List<PintrestItem>
List<PintrestItem>items=new ArrayList<PintrestItem>
items.add(new PintrestItem("你想显示的图片的URL","图片名称"));
items.add(new PintrestItem("你想显示的图片的URL","图片名称"));
items.add(new PintrestItem("你想显示的图片的URL","图片名称"));
recyclerView.setAdapter(new PintrestAdapter(items,getContext()));

```

别忘了在你的 build.gradle 文件中添加 Picasso 依赖：

```

public class PintrestAdapter extends
RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem>images;
Picasso picasso;
Context context;
public PintrestAdapter(ArrayList<PintrestItem>images,Context context){
    this.images=images;
    picasso=Picasso.with(context);
    this.context=context;
}

@Override
public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_layout_item,parent,false);
    return new PintrestViewHolder(view);
}

@Override
public void onBindViewHolder(PintrestViewHolder holder, int position) {
    picasso.load(images.get(position).getUrl()).into(holder.imageView);
    holder.tv.setText(images.get(position).getName());
}

@Override
public int getItemCount() {
    return images.size();
}

public class PintrestViewHolder extends RecyclerView.ViewHolder{
    ImageView imageView;
    TextView tv;
    public PintrestViewHolder(View itemView) {
        super(itemView);
    }
    imageView=(ImageView)itemView.findViewById(R.id.imageView);
    tv=(TextView)itemView.findViewById(R.id.name);
}
}

```

5. Instantiate the RecyclerView in your activity or fragment:

```

RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recyclerView);
//Create the instance of StaggeredGridLayoutManager with 2 rows i.e the span count and provide
the orientation
StaggeredGridLayoutManager layoutManager=new new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
// Create Dummy Data and Add to your List<PintrestItem>
List<PintrestItem>items=new ArrayList<PintrestItem>
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
recyclerView.setAdapter(new PintrestAdapter(items,getContext()));

```

Don't forgot to add the Picasso dependency in your build.gradle file:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

第20章：RecyclerView中的分页

分页是许多需要处理数据列表的移动应用中常见的问题。现在大多数移动应用开始采用“无限分页”模式，滚动时会自动加载新内容。

CWAC Endless Adapter 使在 Android 应用中使用这种模式变得非常简单

第20.1节：MainActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView mRecyclerView;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;

    protected Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pageNumber = 1;
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        tvEmptyView = (TextView) findViewById(R.id.empty_view);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        studentList = new ArrayList<>();
        mTempCheck=new ArrayList<>();
    }
}
```

Chapter 20: Pagination in RecyclerView

Pagination is a common issue with for a lot of mobile apps that need to deal with lists of data. Most of the mobile apps are now starting to take up the "endless page" model, where scrolling automatically loads in new content. CWAC Endless Adapter makes it really easy to use this pattern in Android applications

Section 20.1: MainActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView m.RecyclerView;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;

    protected Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pageNumber = 1;
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        tvEmptyView = (TextView) findViewById(R.id.empty_view);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        studentList = new ArrayList<>();
        mTempCheck=new ArrayList<>();
    }
}
```

```

handler = new Handler();
    if (toolbar != null) {
setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Android 学生");
    }

mRecyclerView.setHasFixedSize(true);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new DataAdapter(studentList, mRecyclerView);
    mRecyclerView.setAdapter(mAdapter);
GetGroupData(" " + mStart, " " + mEnd);
    mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
        @Override
        public void onLoadMore() {
            if( mTempCheck.size()> 0) {
studentList.add(null);
                mAdapter.notifyItemInserted(studentList.size() - 1);
                int start = pageNumber * 20;
                start = start + 1;
                ++ pageNumber;
mTempCheck.clear();
                GetData(" " + start," "+ mEnd);
            }
        }
    });
}

public void GetData(final String LimitStart, final String LimitEnd) {
    Map<String, String> params = new HashMap<>();
    params.put("LimitStart", LimitStart);
params.put("Limit", LimitEnd);
Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.POST,
    "Your php file link", params,
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            Log.d("ResponseSuccess",response.toString());
            // 处理来自服务的数据
        }
    }, new Response.ErrorListener() {

        @Override
        public void onErrorResponse(VolleyError error) {
            VolleyLog.d("ResponseErrorVolly: " + error.getMessage());
        }
    });
}

// 加载初始数据
private void loadData(int start,int end,boolean notifyadapter) {
    for (int i = start; i <= end; i++) {
studentList.add(new Student("学生 " + i, "androidstudent" + i + "@gmail.com"));
        if(notifyadapter)
mAdapter.notifyItemInserted(studentList.size());
    }
}

```

OnLoadMoreListener.java

```

handler = new Handler();
if (toolbar != null) {
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle("Android Students");
}

mRecyclerView.setHasFixedSize(true);
mLayoutManager = new LinearLayoutManager(this);
mRecyclerView.setLayoutManager(mLayoutManager);
mAdapter = new DataAdapter(studentList, mRecyclerView);
mRecyclerView.setAdapter(mAdapter);
GetGroupData(" " + mStart, " " + mEnd);
mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
    @Override
    public void onLoadMore() {
        if( mTempCheck.size()> 0) {
            studentList.add(null);
            mAdapter.notifyItemInserted(studentList.size() - 1);
            int start = pageNumber * 20;
            start = start + 1;
            ++ pageNumber;
            mTempCheck.clear();
            GetData(" " + start," "+ mEnd);
        }
    }
});

public void GetData(final String LimitStart, final String LimitEnd) {
    Map<String, String> params = new HashMap<>();
    params.put("LimitStart", LimitStart);
    params.put("Limit", LimitEnd);
    Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.POST,
        "Your php file link", params,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                Log.d("ResponseSuccess",response.toString());
                // handle the data from the servoce
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.d("ResponseErrorVolly: " + error.getMessage());
            }
        });
}

// load initial data
private void loadData(int start,int end,boolean notifyadapter) {
    for (int i = start; i <= end; i++) {
        studentList.add(new Student("Student " + i, "androidstudent" + i + "@gmail.com"));
        if(notifyadapter)
            mAdapter.notifyItemInserted(studentList.size());
    }
}

```

OnLoadMoreListener.java

```
public interface OnLoadMoreListener { void onLoadMore(); }
```

DataAdapter.java

```

import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class 数据适配器 extends RecyclerView.Adapter {
    private final int VIEW_ITEM = 1;
    private final int VIEW_PROG = 0;

    private List<学生> 学生列表;

    // 当前滚动位置下方加载更多之前的最小项目数
    // before loading more.
    private int 可见阈值 = 5;
    private int 最后可见项目, 总项目数;
    private boolean 加载中;
    private OnLoadMoreListener 加载更多监听器;

    public 数据适配器(List<学生> 学生们, RecyclerView recyclerView) {
        学生列表 = 学生们;
        if (recyclerView.getLayoutManager() instanceof LinearLayoutManager) {
            final LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
            recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
                @Override
                public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                    super.onScrolled(recyclerView, dx, dy);
                    totalItemCount = linearLayoutManager.getItemCount();
                    lastVisibleItem = linearLayoutManager.findLastVisibleItemPosition();
                    if (!loading && totalItemCount <= (lastVisibleItem +
visibleThreshold)) {
                        if ( onLoadMoreListener != null) {
                            onLoadMoreListener.onLoadMore();
                        }
                        loading = true;
                    }
                }
            });
        }
    }

    @Override
    public int getItemViewType(int position) {

        return studentList.get(position) != null ? VIEW_ITEM : VIEW_PROG;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

```

```
public interface OnLoadMoreListener { void onLoadMore(); }
```

DataAdapter.java

```

import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class DataAdapter extends RecyclerView.Adapter {
    private final int VIEW_ITEM = 1;
    private final int VIEW_PROG = 0;

    private List<Student> studentList;

    // The minimum amount of items to have below your current scroll position
    // before loading more.
    private int visibleThreshold = 5;
    private int lastVisibleItem, totalItemCount;
    private boolean loading;
    private OnLoadMoreListener onLoadMoreListener;

    public DataAdapter(List<Student> students, RecyclerView recyclerView) {
        studentList = students;
        if (recyclerView.getLayoutManager() instanceof LinearLayoutManager) {
            final LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
            recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
                @Override
                public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                    super.onScrolled(recyclerView, dx, dy);
                    totalItemCount = linearLayoutManager.getItemCount();
                    lastVisibleItem = linearLayoutManager.findLastVisibleItemPosition();
                    if (!loading && totalItemCount <= (lastVisibleItem +
visibleThreshold)) {
                        if (onLoadMoreListener != null) {
                            onLoadMoreListener.onLoadMore();
                        }
                        loading = true;
                    }
                }
            });
        }
    }

    @Override
    public int getItemViewType(int position) {

        return studentList.get(position) != null ? VIEW_ITEM : VIEW_PROG;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

```

```

RecyclerView.ViewHolder vh;
    if (viewType == VIEW_ITEM) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_row, parent,
false);
        vh = new StudentViewHolder(v);
    } else {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.progress_item,
parent, false);
        vh = new ProgressViewHolder(v);
    }
    return vh;
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    if (holder instanceof StudentViewHolder) {
        Student singleStudent=studentList.get(position);
        ((StudentViewHolder) holder).tvName.setText(singleStudent.getName());
        ((StudentViewHolder) holder).tvEmailId.setText(singleStudent.getEmailId());
        ((StudentViewHolder) holder).student= singleStudent;
    } 否则 {
        ((ProgressViewHolder) holder).progressBar.setIndeterminate(true);
    }
}

public void setLoaded(boolean state) {
    loading = state;
}

@Override
public int getItemCount() {
    return studentList.size();
}

public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener) {
    this.onLoadMoreListener = onLoadMoreListener;
}

//  

public static class StudentViewHolder extends RecyclerView.ViewHolder {
    public TextView tvName;

    public TextView tvEmailId;

    public Student student;

    public StudentViewHolder(View v) {
        super(v);
        tvName = (TextView) v.findViewById(R.id.tvName);
        tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);
    }
}

public static class ProgressViewHolder extends RecyclerView.ViewHolder {
    public ProgressBar progressBar;

    public ProgressViewHolder(View v) {
        super(v);
        progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
    }
}

```

```

RecyclerView.ViewHolder vh;
    if (viewType == VIEW_ITEM) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_row, parent,
false);
        vh = new StudentViewHolder(v);
    } else {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.progress_item,
parent, false);
        vh = new ProgressViewHolder(v);
    }
    return vh;
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    if (holder instanceof StudentViewHolder) {
        Student singleStudent=studentList.get(position);
        ((StudentViewHolder) holder).tvName.setText(singleStudent.getName());
        ((StudentViewHolder) holder).tvEmailId.setText(singleStudent.getEmailId());
        ((StudentViewHolder) holder).student= singleStudent;
    } else {
        ((ProgressViewHolder) holder).progressBar.setIndeterminate(true);
    }
}

public void setLoaded(boolean state) {
    loading = state;
}

@Override
public int getItemCount() {
    return studentList.size();
}

public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener) {
    this.onLoadMoreListener = onLoadMoreListener;
}

//  

public static class StudentViewHolder extends RecyclerView.ViewHolder {
    public TextView tvName;

    public TextView tvEmailId;

    public Student student;

    public StudentViewHolder(View v) {
        super(v);
        tvName = (TextView) v.findViewById(R.id.tvName);
        tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);
    }
}

public static class ProgressViewHolder extends RecyclerView.ViewHolder {
    public ProgressBar progressBar;

    public ProgressViewHolder(View v) {
        super(v);
        progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
    }
}

```

```
    }  
}
```

```
}
```

```
    }  
}
```

```
}
```

第21章：ImageView

参数

资源ID 描述
你在res文件夹中的图片文件名（通常在drawable文件夹中）

ImageView (`android.widget.ImageView`) 是用于显示和操作图像资源（如 Drawables 和 Bitmaps）的视图。

本主题中讨论的一些效果可以应用于图像。图像源可以在 XML 文件（layout 文件夹）中设置，也可以通过 Java 代码以编程方式设置。

第21.1节：设置色调

为图像设置色调颜色。默认情况下，色调将使用 SRC_ATOP 模式进行混合。

使用 XML 属性设置色调：

```
android:tint="#009c38"
```

注意：必须是颜色值，格式为 "#rgb"、"#argb"、"#rrggbb" 或 "#aarrggbb"。

以编程方式设置色调：

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

你可以清除这个颜色滤镜：

```
imgExample.clearColorFilter();
```

示例：

Normal Image



Image with tint



Chapter 21: ImageView

Parameter

resId Description
your Image file name in the res folder (usually in drawable folder)

ImageView (`android.widget.ImageView`) is a View for displaying and manipulating image resources, such as Drawables and Bitmaps.

Some effects, discussed in this topic, can be applied to the image. The image source can be set in XML file (layout folder) or by programmatically in Java code.

Section 21.1: Set tint

Set a tinting color for the image. By default, the tint will blend using SRC_ATOP mode.

set tint using XML attribute:

```
android:tint="#009c38"
```

Note: Must be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

set tint programmatically:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

and you can clear this color filter:

```
imgExample.clearColorFilter();
```

Example:

Normal Image



Image with tint



第21.2节：设置透明度

"alpha"用于指定图像的不透明度。

使用XML属性设置透明度：

```
android:alpha="0.5"
```

注意：取值为0（完全透明）到1（完全可见）的浮点数

通过代码设置透明度：

```
imgExample.setAlpha(0.5f);
```

Normal Image



Image with alpha



Section 21.2: Set alpha

"alpha" is used to specify the opacity for an image.

set alpha using XML attribute:

```
android:alpha="0.5"
```

Note: takes float value from 0 (transparent) to 1 (fully visible)

set alpha programmatically:

```
imgExample.setAlpha(0.5f);
```

Normal Image



Image with alpha



第21.3节：设置缩放类型

控制图像应如何调整大小或移动以匹配ImageView的大小。

Section 21.3: Set Scale Type

Controls how the image should be resized or moved to match the size of ImageView.

XML 属性：

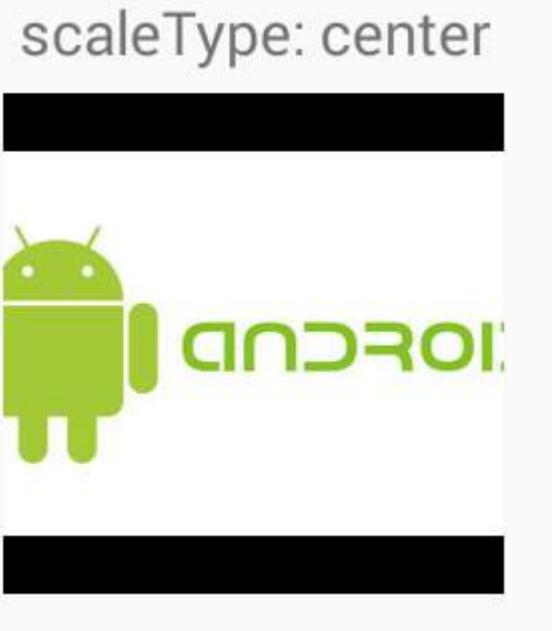
```
android:scaleType="..."
```

我将用一个带有黑色背景的正方形ImageView来说明不同的缩放类型，我们想在ImageView中显示一个白色背景的矩形可绘制对象。

```
<ImageView  
    android:id="@+id/imgExample"  
    android:layout_width="200dp"  
    android:layout_height="200dp"  
    android:background="#000"  
    android:src="@drawable/android2"  
    android:scaleType="..."/>
```

scaleType 必须是以下值之一：

1. center: 将图像居中显示在视图中，但不进行缩放。



2. centerCrop: 均匀缩放图像（保持图像的宽高比），使图像的两个维度（宽度和高度）都等于或大于视图的对应维度（减去内边距）。

然后图像在视图中居中显示。

XML attribute:

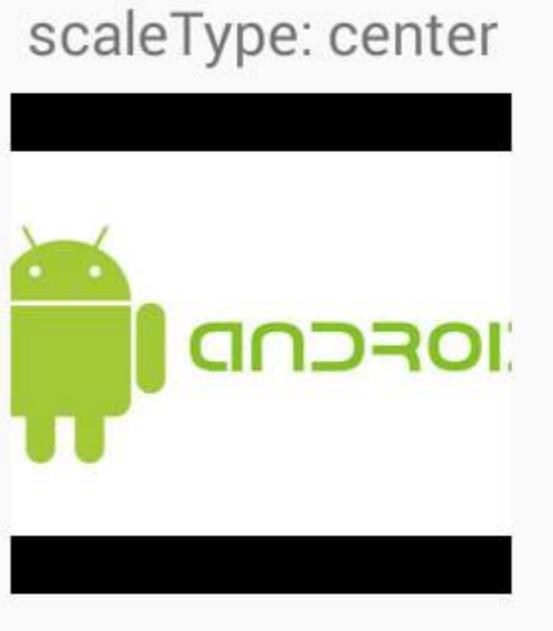
```
android:scaleType="..."
```

i will illustrate different scale types with a square ImageView which has a black background and we want to display a rectangular drawable in white background in ImageView.

```
<ImageView  
    android:id="@+id/imgExample"  
    android:layout_width="200dp"  
    android:layout_height="200dp"  
    android:background="#000"  
    android:src="@drawable/android2"  
    android:scaleType="..."/>
```

scaleType must be one of the following values:

1. center:Center the image in the view, but perform no scaling.



2. centerCrop: Scale the image uniformly (maintain the image's aspect ratio) so both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding). The image is then centered in the view.

scaleType: centerCrop



3. centerInside : 均匀缩放图像（保持图像的宽高比），使图像的两个维度（宽度和高度）都等于或小于视图的对应维度（减去内边距）。然后图像在视图中居中。

scaleType: centerInside



4. matrix : 绘制时使用图像矩阵进行缩放。

scaleType: centerCrop



3. centerInside: Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding). The image is then centered in the view.

scaleType: centerInside



4. matrix : Scale using the image matrix when drawing.

scaleType: matrix



5. fitXY : 使用FILL缩放图像。

scaleType: fitXY



6. fitStart : 使用START缩放图像。

scaleType: matrix



5. fitXY: Scale the image using [FILL](#).

scaleType: fitXY



6. fitStart: Scale the image using [START](#).

scaleType: fitStart



7. fitCenter : 使用CENTER缩放图像。_____

scaleType: fitCenter



8. fitEnd : 使用END缩放图像。_____

scaleType: fitStart



7. fitCenter: Scale the image using [CENTER](#).

scaleType: fitCenter



8. fitEnd: Scale the image using [END](#).

scaleType: fitEnd



第21.4节：ImageView 缩放类型 - 居中

ImageView 中包含的图像可能无法完全适应容器所给定的尺寸。在这种情况下，框架允许你以多种方式调整图像大小。

居中

```
<ImageView android:layout_width="20dp"
    android:layout_height="20dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="center"
    android:background="@android:color/holo_orange_light"/>
```

这不会调整图像大小，而是将其居中显示在容器内（橙色 = 容器）

scaleType: fitEnd



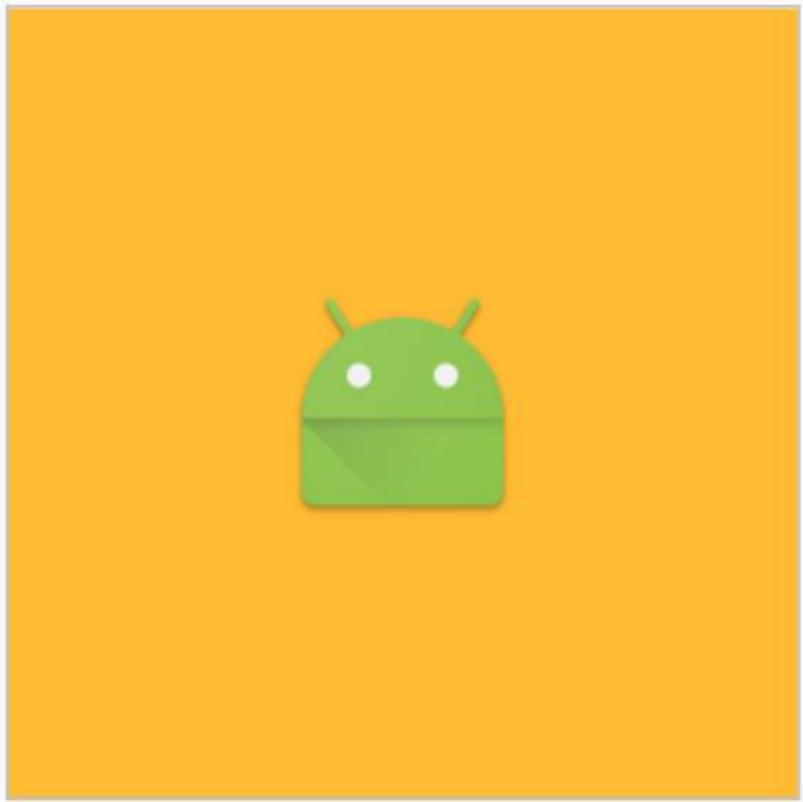
Section 21.4: ImageView ScaleType - Center

The image contained in the ImageView may not fit the exact size given to the container. In that case, the framework allows you to resize the image in a number of ways.

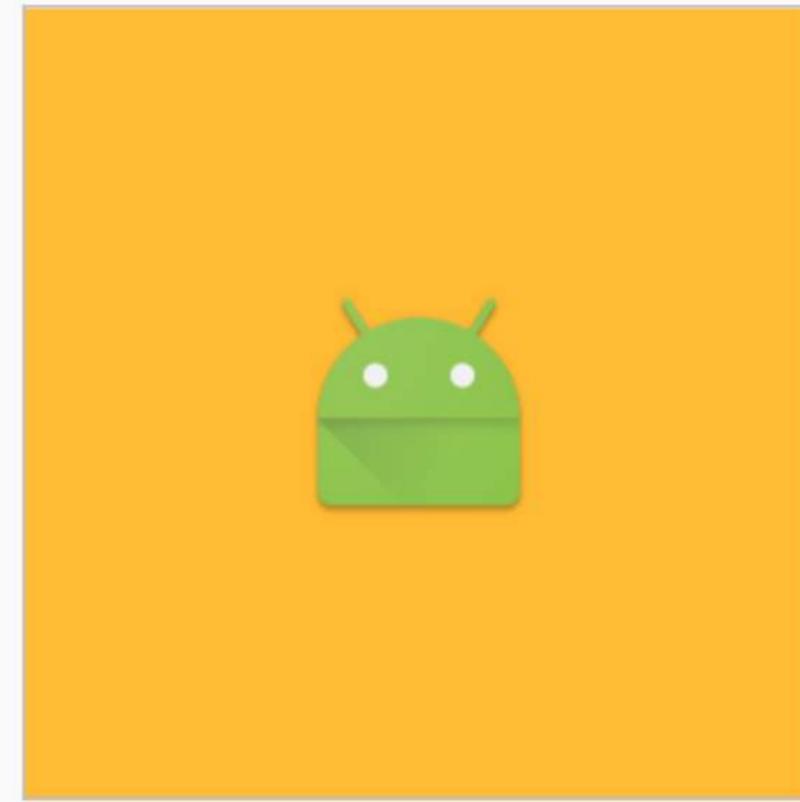
Center

```
<ImageView android:layout_width="20dp"
    android:layout_height="20dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="center"
    android:background="@android:color/holo_orange_light"/>
```

This will not resize the image, and it will center it inside the container (Orange = container)



如果 ImageView 比图像小，图像不会被缩放，你只能看到图像的一部分



In case that the ImageView is smaller than the image, the image will not be resized and you will only be able to see a part of it



加粗文本

第21.5节：ImageView 缩放类型 - CenterCrop

均匀缩放图像（保持图像的宽高比），使图像的两个维度（宽度和高度）都等于或大于视图的相应维度（减去内边距）。

[官方文档](#)

当图像与容器的比例相同时：



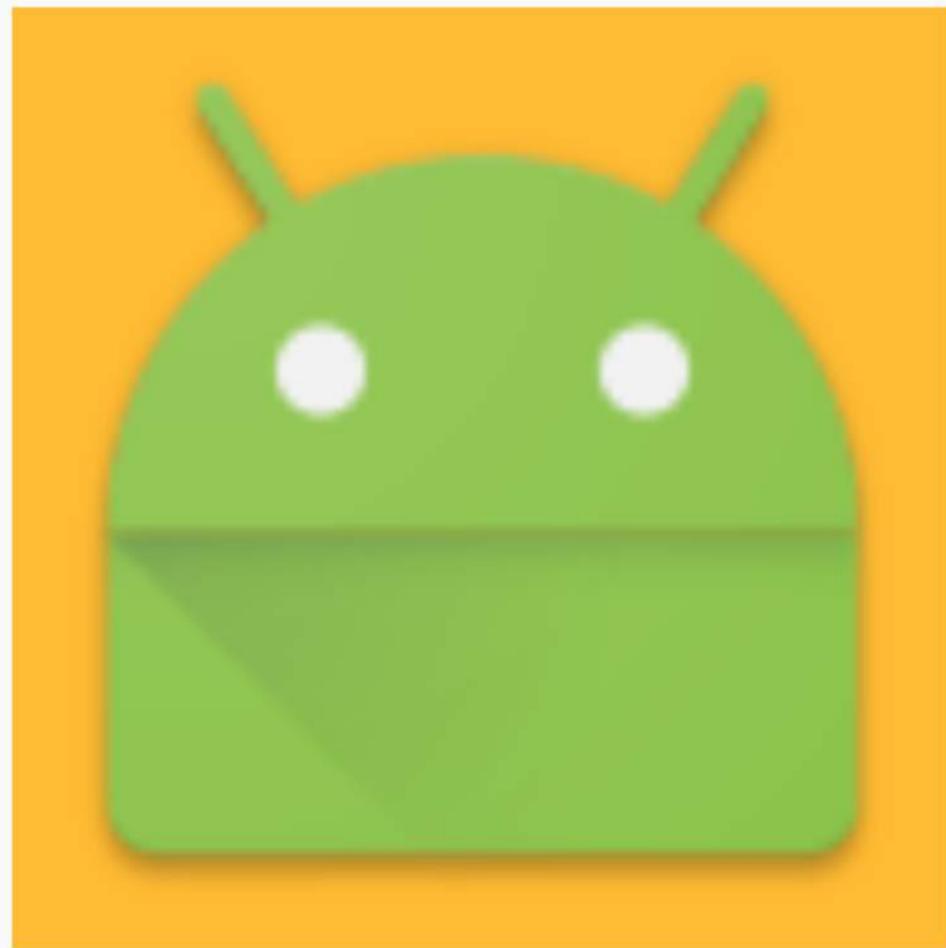
strong text

Section 21.5: ImageView ScaleType - CenterCrop

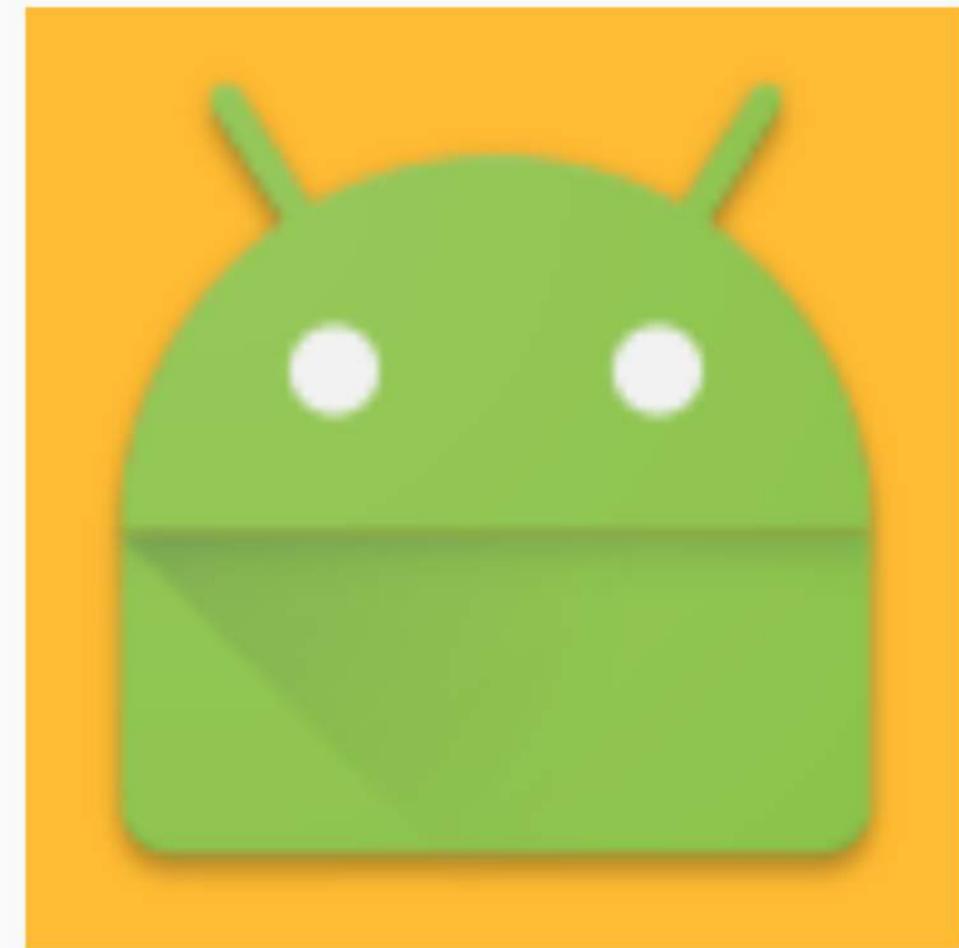
Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding).

[Official Docs](#)

When the image matches the proportions of the container:



当图像比容器宽时，它会将图像扩展到较大的尺寸（此处为高度），并调整图像的宽度而不改变其比例，导致图像被裁剪。



When the image is wider than the container it will expand it to the bigger size (in this case height) and adjust the width of the image without changing its proportions, causing it to crop.



第21.6节：ImageView 缩放类型 - CenterInside

均匀缩放图像（保持图像的宽高比），使图像的两个维度（宽度和高度）都等于或小于视图的相应维度（减去内边距）。

[官方文档](#)

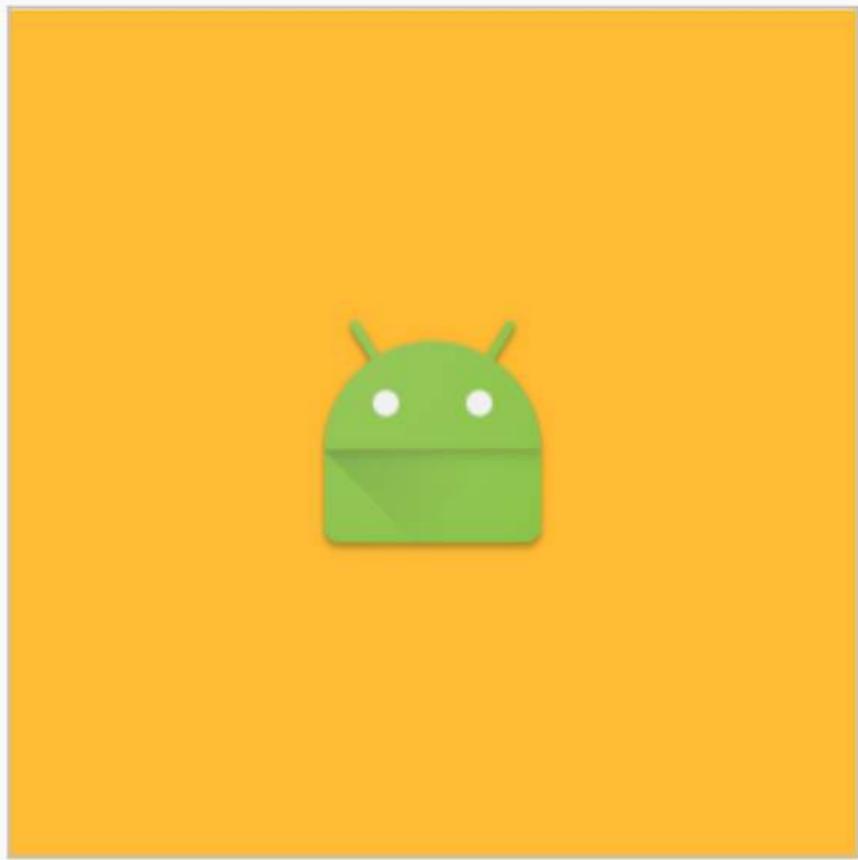
它会将图像居中并调整为较小的尺寸，如果容器的两个尺寸都较大，则表现与 center 相同。

Section 21.6: ImageView ScaleType - CenterInside

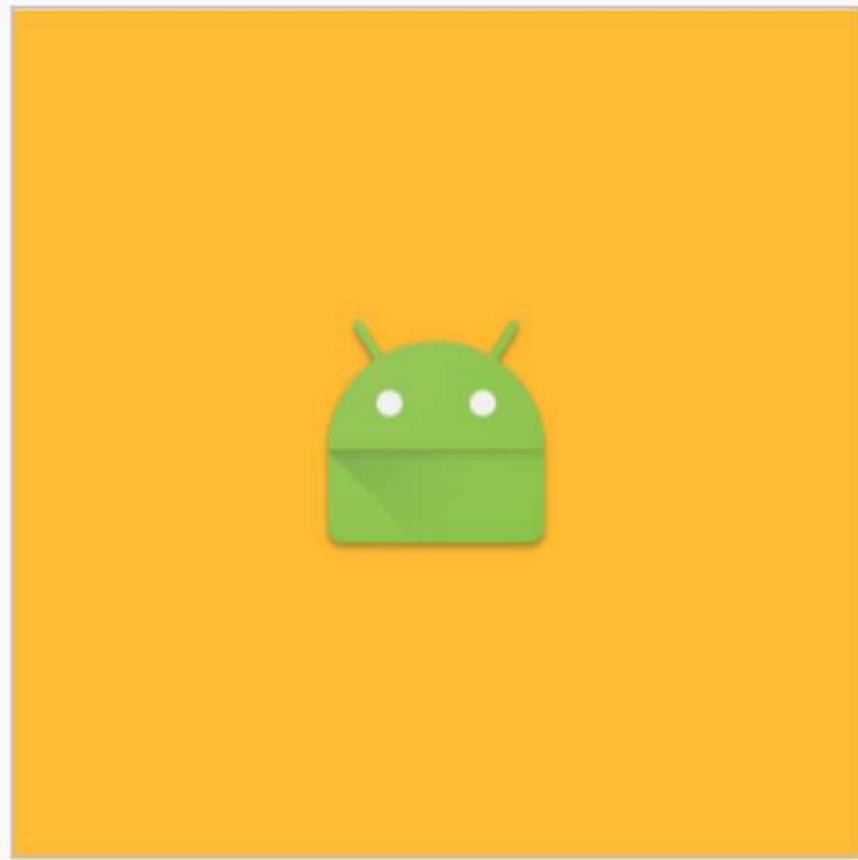
Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding).

[Official Docs](#)

It will center the image and resize it to the smaller size, if both container sizes are bigger it will act the same as center.



但如果其中一个尺寸较小，它将适合该尺寸。



But if one of the sizes are small, it will fit to that size.



第21.7节：ImageView 缩放类型 - FitStart 和 FitEnd

使用 START 缩放图像。

使用 END 缩放图像。

[官方文档](#)

FitStart

这将适应容器的最小尺寸，并将其对齐到起始位置。

```
<ImageView android:layout_width="200dp"  
          android:layout_height="200dp"
```

Section 21.7: ImageView ScaleType - FitStart and FitEnd

Scale the image using START.

Scale the image using END.

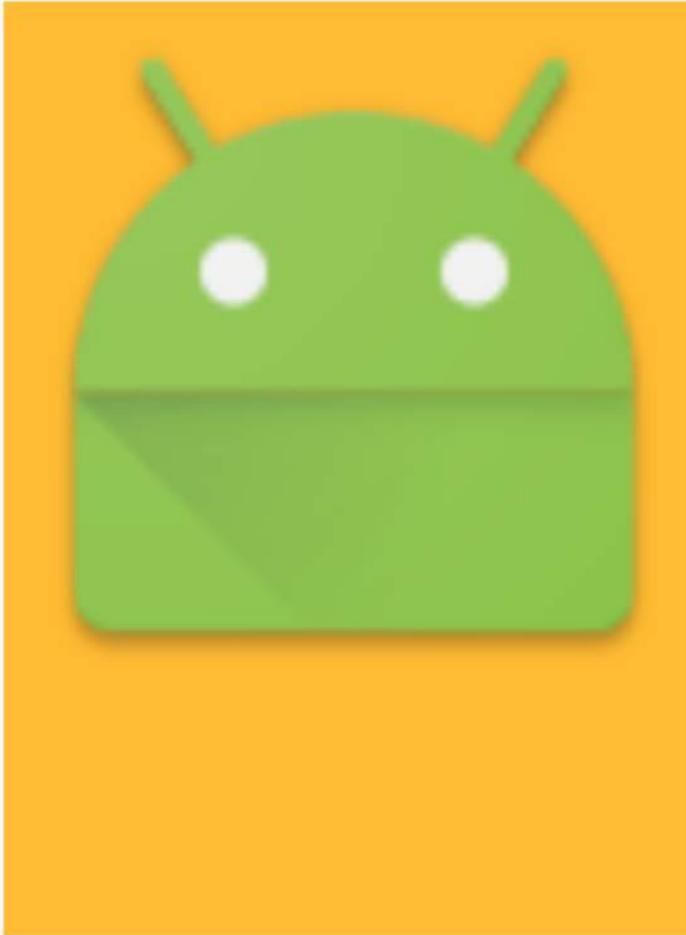
[Official Docs](#)

FitStart

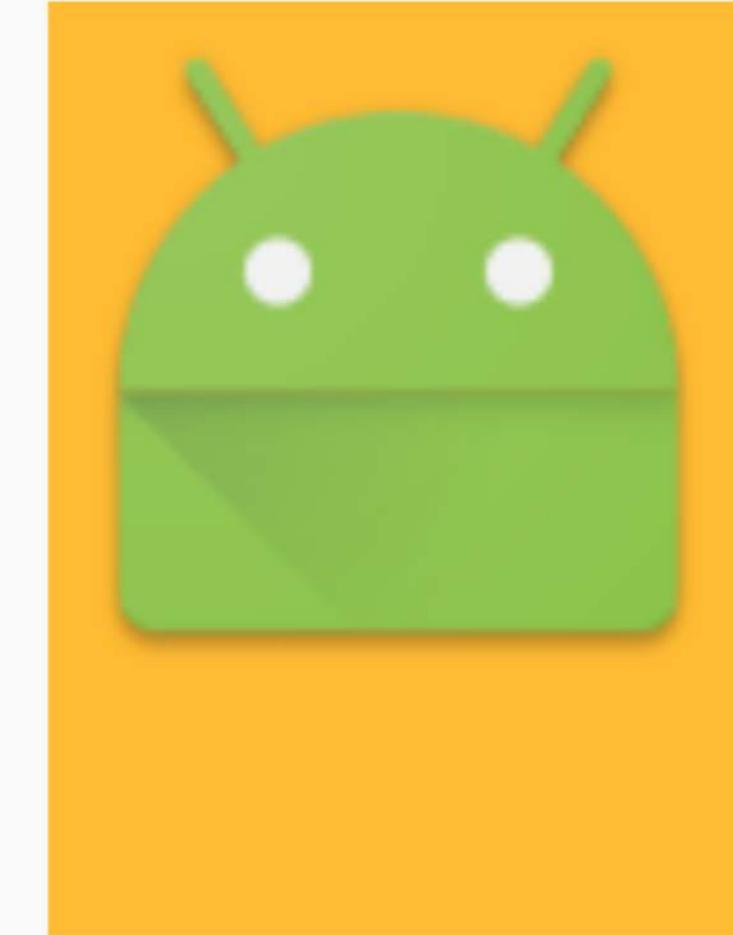
This will fit to the smallest size of the container, and it will align it to the start.

```
<ImageView android:layout_width="200dp"  
          android:layout_height="200dp"
```

```
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitStart"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```



```
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitStart"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```





FitEnd

这将适应容器的最小尺寸，并将其对齐到末端。

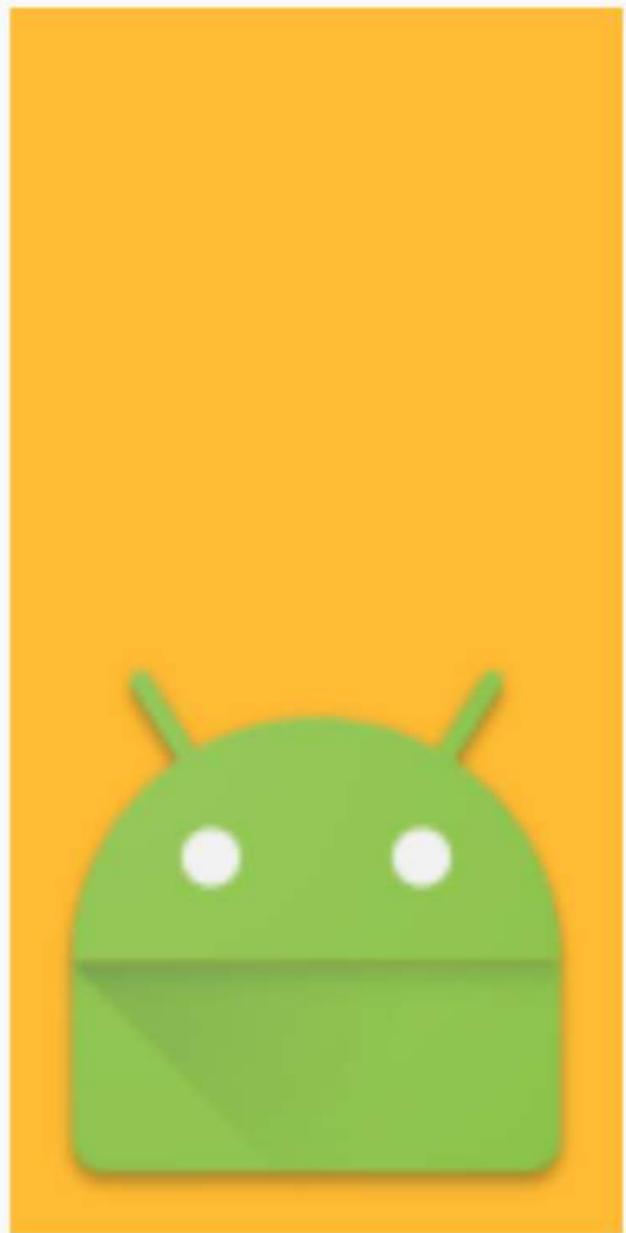
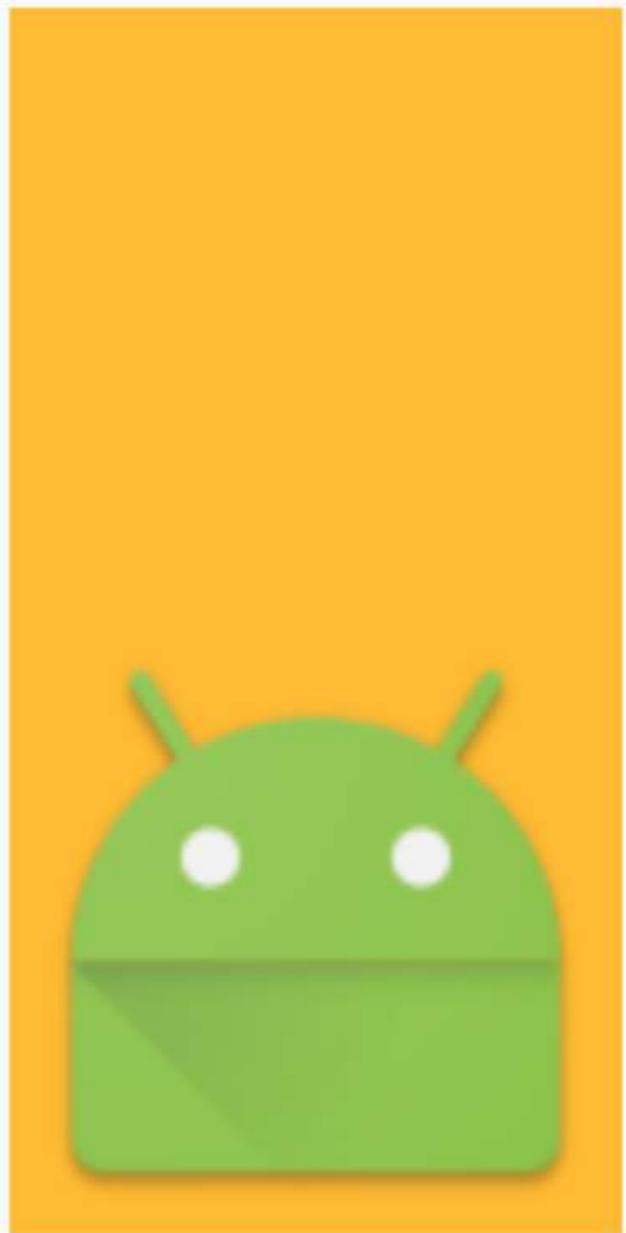
```
<ImageView android:layout_width="200dp"
    android:layout_height="100dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitEnd"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```



FitEnd

This will fit to the smallest size of the container, and it will align it to the end.

```
<ImageView android:layout_width="200dp"
    android:layout_height="100dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitEnd"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```





第21.8节：ImageView ScaleType - FitCenter

使用 CENTER 缩放图像。

[官方文档](#)

这会放大图像以尝试匹配容器，并将其对齐到中心，适应较小的尺寸。

较高的高度（适应宽度）



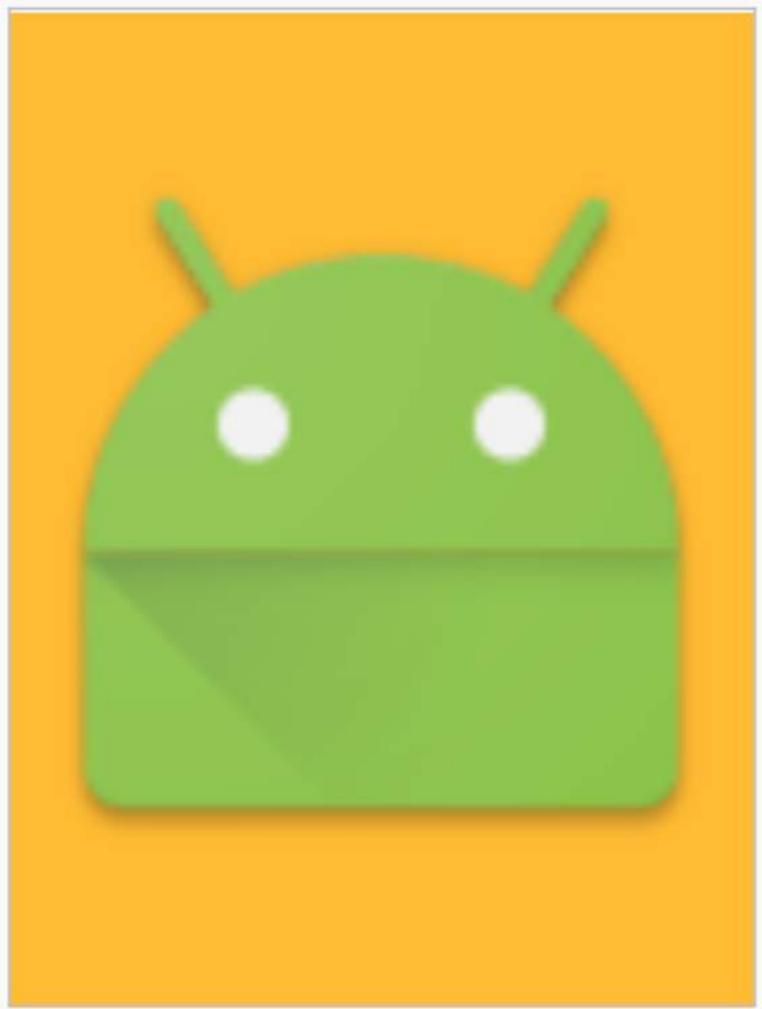
Section 21.8: ImageView ScaleType - FitCenter

Scale the image using CENTER.

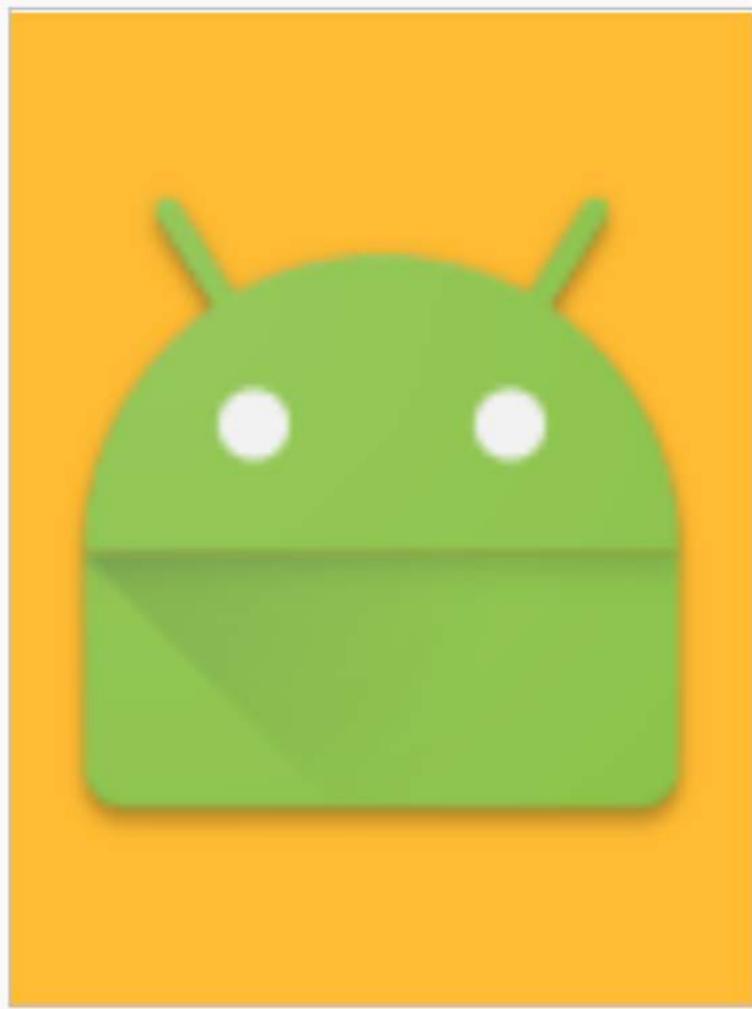
[Official Docs](#)

This expands the image to try to match the container and it will align it to the center, it will fit to the smaller size.

Bigger height (fit to width)



相同的宽度和高度。



Same width and height.



第21.9节：设置图像资源

```
<ImageView  
    android:id="@+id/imgExample"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    ...  
/>
```

使用XML属性将drawable设置为ImageView的内容：

```
    android:src="@drawable/android2"
```

以编程方式设置drawable：

```
ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
```

Section 21.9: Set Image Resource

```
<ImageView  
    android:id="@+id/imgExample"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    ...  
/>
```

set a drawable as content of ImageView using XML attribute:

```
    android:src="@drawable/android2"
```

set a drawable programmatically:

```
ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
```

```
imgExample.setImageResource(R.drawable.android2);
```

第21.10节：ImageView ScaleType - FitXY

使用FILL缩放图像。

[官方文档](#)

```
<ImageView android:layout_width="100dp"
    android:layout_height="200dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitXY"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```



```
imgExample.setImageResource(R.drawable.android2);
```

Section 21.10: ImageView ScaleType - FitXY

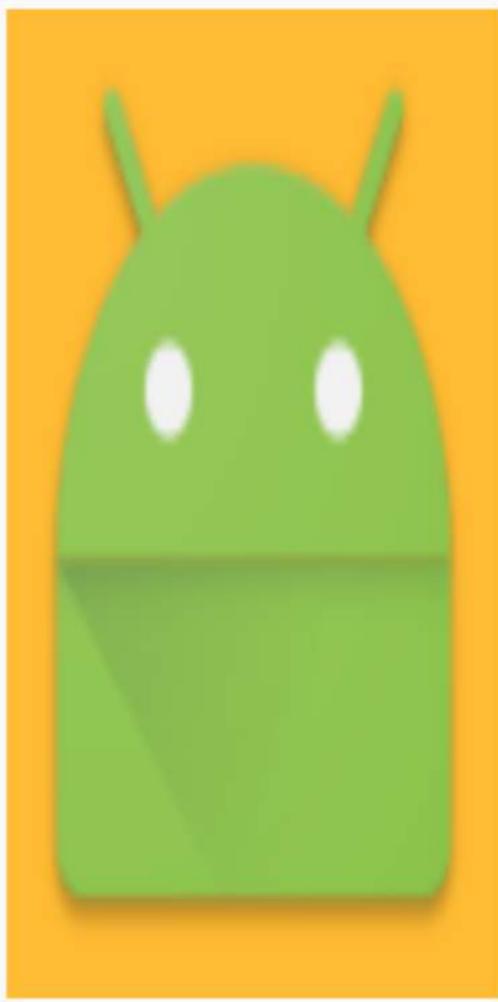
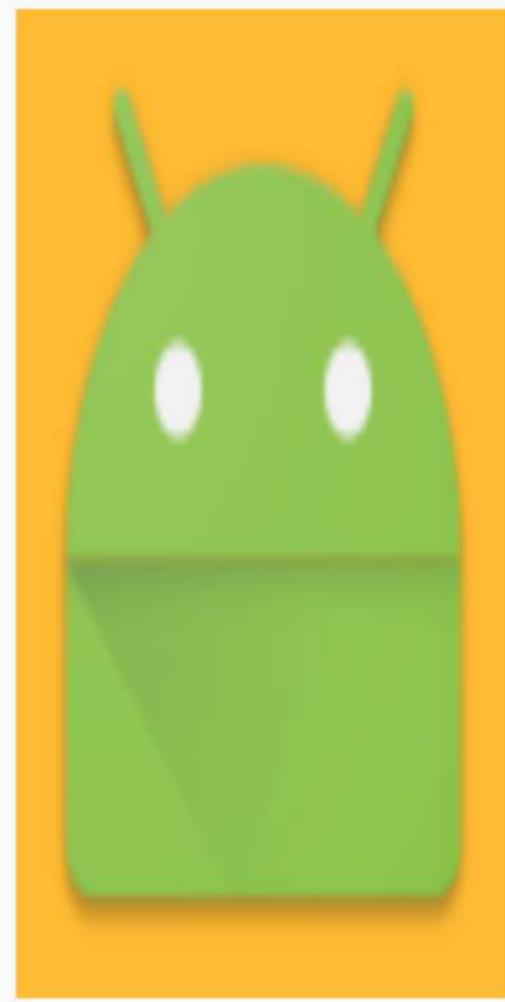
Scale the image using FILL.

[Official Docs](#)

```
<ImageView android:layout_width="100dp"
    android:layout_height="200dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitXY"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```







第21.11节：MLRoundedImageView.java

将以下类复制并粘贴到您的包中：

```
public class MLRoundedImageView extends ImageView {  
  
    public MLRoundedImageView(Context context) {  
        super(context);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs, int defStyle) {  
        super(context, attrs, defStyle);  
    }  
}
```

Section 21.11: MLRoundedImageView.java

Copy and Paste following class in your package:

```
public class MLRoundedImageView extends ImageView {  
  
    public MLRoundedImageView(Context context) {  
        super(context);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs, int defStyle) {  
        super(context, attrs, defStyle);  
    }  
}
```

```

@Override
protected void onDraw(Canvas canvas) {

    Drawable drawable = getDrawable();

    if (drawable == null) {
        return;
    }

    if (getWidth() == 0 || getHeight() == 0) {
        return;
    }
    Bitmap b = ((BitmapDrawable) drawable).getBitmap();
    Bitmap bitmap = b.copy(Bitmap.Config.ARGB_8888, true);

    int w = getWidth(), h = getHeight();

    Bitmap roundBitmap = getCroppedBitmap(bitmap, w);
    canvas.drawBitmap(roundBitmap, 0, 0, null);
}

public static Bitmap getCroppedBitmap(Bitmap bmp, int radius) {
    Bitmap sbmp;

    if (bmp.getWidth() != radius || bmp.getHeight() != radius) {
        float smallest = Math.min(bmp.getWidth(), bmp.getHeight());
        float factor = smallest / radius;
        sbmp = Bitmap.createScaledBitmap(bmp, (int)(bmp.getWidth() / factor),
            (int)(bmp.getHeight() / factor), false);
    } 否则 {
        sbmp = bmp;
    }

    Bitmap output = Bitmap.createBitmap(radius, radius,
        Config.ARGB_8888);
    Canvas canvas = new Canvas(output);

    final int color = 0xffa19774;
    final Paint paint = new Paint();
    final Rect rect = new Rect(0, 0, radius, radius);

    paint.setAntiAlias(true);
    paint.setFilterBitmap(true);
    paint.setDither(true);
    canvas.drawARGB(0, 0, 0, 0);
    paint.setColor(Color.parseColor("#BAB399"));
    canvas.drawCircle(radius / 2 + 0.7f,
        radius / 2 + 0.7f, radius / 2 + 0.1f, paint);
    paint.setXfermode(new PorterDuffXfermode_Mode.SRC_IN));
    canvas.drawBitmap(sbmp, rect, rect, paint);

    return output;
}
}

```

在XML中使用此类时，请使用包名替代ImageView

```

<com.androidbutts.example.MLRoundedImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

@Override
protected void onDraw(Canvas canvas) {

    Drawable drawable = getDrawable();

    if (drawable == null) {
        return;
    }

    if (getWidth() == 0 || getHeight() == 0) {
        return;
    }
    Bitmap b = ((BitmapDrawable) drawable).getBitmap();
    Bitmap bitmap = b.copy(Bitmap.Config.ARGB_8888, true);

    int w = getWidth(), h = getHeight();

    Bitmap roundBitmap = getCroppedBitmap(bitmap, w);
    canvas.drawBitmap(roundBitmap, 0, 0, null);
}

public static Bitmap getCroppedBitmap(Bitmap bmp, int radius) {
    Bitmap sbmp;

    if (bmp.getWidth() != radius || bmp.getHeight() != radius) {
        float smallest = Math.min(bmp.getWidth(), bmp.getHeight());
        float factor = smallest / radius;
        sbmp = Bitmap.createScaledBitmap(bmp, (int)(bmp.getWidth() / factor),
            (int)(bmp.getHeight() / factor), false);
    } else {
        sbmp = bmp;
    }

    Bitmap output = Bitmap.createBitmap(radius, radius,
        Config.ARGB_8888);
    Canvas canvas = new Canvas(output);

    final int color = 0xffa19774;
    final Paint paint = new Paint();
    final Rect rect = new Rect(0, 0, radius, radius);

    paint.setAntiAlias(true);
    paint.setFilterBitmap(true);
    paint.setDither(true);
    canvas.drawARGB(0, 0, 0, 0);
    paint.setColor(Color.parseColor("#BAB399"));
    canvas.drawCircle(radius / 2 + 0.7f,
        radius / 2 + 0.7f, radius / 2 + 0.1f, paint);
    paint.setXfermode(new PorterDuffXfermode_Mode.SRC_IN));
    canvas.drawBitmap(sbmp, rect, rect, paint);

    return output;
}
}

```

Use this Class in XML with package name instead of ImageView

```

<com.androidbutts.example.MLRoundedImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

android:src="@mipmap/ic_launcher" />

android:src="@mipmap/ic_launcher" />

第22章：VideoView

第22.1节：使用VideoView从URL播放视频

```
videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
videoView.requestFocus();

videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
    }
});

videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
        videoView.start();
    }
});

mediaPlayer.setOnVideoSizeChangedListener(new MediaPlayer.OnVideoSizeChangedListener() {
    @Override
    public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
        MediaController mediaController = new MediaController(ActivityName.this);
        videoView.setMediaController(mediaController);
    }
});

mediaController.setAnchorView(videoView);

videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
        return false;
    }
});
```

第22.2节：创建VideoView

在Activity中找到VideoView并添加视频。

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

开始播放视频。

```
videoView.start();
```

在XML布局文件中定义VideoView。

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Chapter 22: VideoView

Section 22.1: Play video from URL with using VideoView

```
videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
videoView.requestFocus();

videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
    }
});

videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
        videoView.start();
        mediaPlayer.setOnVideoSizeChangedListener(new MediaPlayer.OnVideoSizeChangedListener() {
            @Override
            public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
                MediaController mediaController = new MediaController(ActivityName.this);
                videoView.setMediaController(mediaController);
                mediaController.setAnchorView(videoView);
            }
        });
    }
});

videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
        return false;
    }
});
```

Section 22.2: VideoView Create

Find VideoView in Activity and add video into it.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Start playing video.

```
videoView.start();
```

Define VideoView in XML Layout file.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

第23章：优化的视频视图（VideoView）

使用继承自SurfaceView的VideoView在ListView的一行中播放视频，起初似乎可以正常工作，直到用户尝试滚动列表。一旦列表开始滚动，视频就会变黑（有时显示为白色）。视频仍在后台播放，但你看不到了，因为它将视频的其余部分渲染为一个黑色框。使用自定义的优化视频视图，视频将在ListView滚动时正常播放，就像我们的Instagram、Facebook、Twitter。

23.1节：ListView中的优化视频视图

这是你需要放在包中的自定义VideoView。

自定义视频视图布局：

```
<your.packagename.VideoView  
    android:id="@+id/video_view"  
    android:layout_width="300dp"  
    android:layout_height="300dp" />
```

自定义优化VideoView的代码：

```
package your.package.com.whateveritis;  
  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.SurfaceTexture;  
import android.media.AudioManager;  
import android.media.MediaPlayer;  
import android.media.MediaPlayer.OnCompletionListener;  
import android.media.MediaPlayer.OnErrorListener;  
import android.media.MediaPlayer.OnInfoListener;  
import android.net.Uri;  
import android.util.AttributeSet;  
import android.util.Log;  
import android.view.KeyEvent;  
import android.view.MotionEvent;  
import android.view.Surface;  
import android.view.TextureView;  
import android.view.View;  
import android.widget.MediaController;  
import android.widget.MediaController.MediaPlayerControl;  
  
import java.io.IOException;  
  
/**  
 * VideoView 用于播放视频，就像 * {@link android.widget.  
 * VideoView VideoView}。我们定义了一个自定义视图，因为 * 我们无法在 ListView 中使用 {  
 * {@link android.widget.VideoView VideoView}}。<br/> * ScrollView 中的 VideoView 无法正常滚  
动。即使你使用 * 解决方法设置背景颜色，MediaController 也不会随着 VideoView 一起滚动。  
此外， * 使用该解决方法时，滚动的视频效果非常糟糕，闪烁严重。  
*  
* 作者 leo  
*/  
public class VideoView extends TextureView implements MediaPlayerControl {
```

Chapter 23: Optimized VideoView

Playing a video using a VideoView which extends SurfaceView inside of a row of a ListView seems to work at first, until the user tries to scroll the list. As soon as the list starts to scroll, the video turns black (sometimes displays white). It keeps playing in the background but you can't see it anymore because it renders the rest of the video as a black box. With the custom Optimized VideoView, the videos will play on scroll in the ListView just like our Instagram, Facebook, Twitter.

Section 23.1: Optimized VideoView in ListView

This is the custom VideoView that you need to have it in your package.

Custom VideoView Layout:

```
<your.packagename.VideoView  
    android:id="@+id/video_view"  
    android:layout_width="300dp"  
    android:layout_height="300dp" />
```

Code for custom Optimized VideoView:

```
package your.package.com.whateveritis;  
  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.SurfaceTexture;  
import android.media.AudioManager;  
import android.media.MediaPlayer;  
import android.media.MediaPlayer.OnCompletionListener;  
import android.media.MediaPlayer.OnErrorListener;  
import android.media.MediaPlayer.OnInfoListener;  
import android.net.Uri;  
import android.util.AttributeSet;  
import android.util.Log;  
import android.view.KeyEvent;  
import android.view.MotionEvent;  
import android.view.Surface;  
import android.view.TextureView;  
import android.view.View;  
import android.widget.MediaController;  
import android.widget.MediaController.MediaPlayerControl;  
  
import java.io.IOException;  
  
/**  
 * VideoView is used to play video, just like  
 * {@link android.widget.VideoView VideoView}. We define a custom view, because  
 * we could not use {@link android.widget.VideoView VideoView} in ListView. <br/>  
 * VideoViews inside ScrollView do not scroll properly. Even if you use the  
 * workaround to set the background color, the MediaController does not scroll  
 * along with the VideoView. Also, the scrolling video looks horrendous with the  
 * workaround, lots of flickering.  
*  
* @author leo  
*/  
public class VideoView extends TextureView implements MediaPlayerControl {
```

```

private static final String TAG = "tag";

// 所有可能的内部状态
private static final int STATE_ERROR = -1;
private static final int STATE_IDLE = 0;
private static final int STATE_PREPARING = 1;
private static final int STATE_PREPARED = 2;
private static final int STATE_PLAYING = 3;
private static final int STATE_PAUSED = 4;
private static final int STATE_PLAYBACK_COMPLETED = 5;

// currentState 是 VideoView 对象的当前状态。
// targetState 是方法调用者希望达到的状态。
// 例如，无论 VideoView 对象的当前状态如何，
// 调用 pause() 都是希望将对象带到目标状态
// STATE_PAUSED。
private int mCurrentState = STATE_IDLE;
private int mTargetState = STATE_IDLE;

// 播放和显示视频所需的内容
private MediaPlayer mMediaPlayer;
private int mVideoWidth;
private int mVideoHeight;
private int mSurfaceWidth;
private int mSurfaceHeight;
private SurfaceTexture mSurfaceTexture;
private Surface mSurface;
private MediaController mMediaController;
private MediaPlayer.OnCompletionListener mOnCompletionListener;
private MediaPlayer.OnPreparedListener mOnPreparedListener;

private MediaPlayer.OnErrorListener mOnErrorListener;
private MediaPlayer.OnInfoListener mOnInfoListener;

private int mSeekWhenPrepared; // 记录准备期间的seek位置

private int mCurrentBufferPercentage;
private int mAudioSession;
private Uri mUri;

private Context mContext;

public VideoView(final Context context) {
    super(context);
    mContext = context;
    initVideoView();
}

public VideoView(final Context context, final AttributeSet attrs) {
    super(context, attrs);
    mContext = context;
    initVideoView();
}

public VideoView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    mContext = context;
    initVideoView();
}

public void initVideoView() {
    mVideoHeight = 0;
}

```

```

private static final String TAG = "tag";

// all possible internal states
private static final int STATE_ERROR = -1;
private static final int STATE_IDLE = 0;
private static final int STATE_PREPARING = 1;
private static final int STATE_PREPARED = 2;
private static final int STATE_PLAYING = 3;
private static final int STATE_PAUSED = 4;
private static final int STATE_PLAYBACK_COMPLETED = 5;

// currentState is a VideoView object's current state.
// targetState is the state that a method caller intends to reach.
// For instance, regardless the VideoView object's current state,
// calling pause() intends to bring the object to a target state
// of STATE_PAUSED.
private int mCurrentState = STATE_IDLE;
private int mTargetState = STATE_IDLE;

// Stuff we need for playing and showing a video
private MediaPlayer mMediaPlayer;
private int mVideoWidth;
private int mVideoHeight;
private int mSurfaceWidth;
private int mSurfaceHeight;
private SurfaceTexture mSurfaceTexture;
private Surface mSurface;
private MediaController mMediaController;
private MediaPlayer.OnCompletionListener mOnCompletionListener;
private MediaPlayer.OnPreparedListener mOnPreparedListener;

private MediaPlayer.OnErrorListener mOnErrorListener;
private MediaPlayer.OnInfoListener mOnInfoListener;

private int mSeekWhenPrepared; // recording the seek position while
// preparing
private int mCurrentBufferPercentage;
private int mAudioSession;
private Uri mUri;

private Context mContext;

public VideoView(final Context context) {
    super(context);
    mContext = context;
    initVideoView();
}

public VideoView(final Context context, final AttributeSet attrs) {
    super(context, attrs);
    mContext = context;
    initVideoView();
}

public VideoView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    mContext = context;
    initVideoView();
}

public void initVideoView() {
    mVideoHeight = 0;
}

```

```

mVideoWidth = 0;
setFocusable(false);
setSurfaceTextureListener(mSurfaceTextureListener);
}

public int resolveAdjustedSize(int desiredSize, int measureSpec) {
    int result = desiredSize;
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

    switch (specMode) {
        case MeasureSpec.UNSPECIFIED:
            /*
             * 父容器表示我们可以任意大。只要不要超过
             * 自己施加的最大尺寸即可。
             */
            result = desiredSize;
            break;

        case MeasureSpec.AT_MOST:
            /*
             * 父容器表示我们可以任意大，最大到specSize。不要
             * 超过specSize，也不要超过自己施加的最大尺寸
             */
            result = Math.min(desiredSize, specSize);
            break;

        case MeasureSpec.EXACTLY:
            // 没有选择。按指示执行。
            result = specSize;
            break;
    }
    return result;
}

public void setVideoPath(String path) {
    Log.d(TAG, "设置视频路径为: " + path);
    setVideoURI(Uri.parse(path));
}

public void setVideoURI(Uri _videoURI) {
    mUri = _videoURI;
    mSeekWhenPrepared = 0;
    requestLayout();
    invalidate();
    openVideo();
}

public Uri getUri() {
    return mUri;
}

public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
    mSurfaceTexture = _surfaceTexture;
}

public void openVideo() {
    if ((mUri == null) || (mSurfaceTexture == null)) {
        Log.d(TAG, "无法打开视频，uri 或 surface texture 为空。");
        return;
    }
}

```

```

mVideoWidth = 0;
setFocusable(false);
setSurfaceTextureListener(mSurfaceTextureListener);
}

public int resolveAdjustedSize(int desiredSize, int measureSpec) {
    int result = desiredSize;
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

    switch (specMode) {
        case MeasureSpec.UNSPECIFIED:
            /*
             * Parent says we can be as big as we want. Just don't be larger
             * than max size imposed on ourselves.
             */
            result = desiredSize;
            break;

        case MeasureSpec.AT_MOST:
            /*
             * Parent says we can be as big as we want, up to specSize. Don't be
             * larger than specSize, and don't be larger than the max size
             * imposed on ourselves.
             */
            result = Math.min(desiredSize, specSize);
            break;

        case MeasureSpec.EXACTLY:
            // No choice. Do what we are told.
            result = specSize;
            break;
    }
    return result;
}

public void setVideoPath(String path) {
    Log.d(TAG, "Setting video path to: " + path);
    setVideoURI(Uri.parse(path));
}

public void setVideoURI(Uri _videoURI) {
    mUri = _videoURI;
    mSeekWhenPrepared = 0;
    requestLayout();
    invalidate();
    openVideo();
}

public Uri getUri() {
    return mUri;
}

public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
    mSurfaceTexture = _surfaceTexture;
}

public void openVideo() {
    if ((mUri == null) || (mSurfaceTexture == null)) {
        Log.d(TAG, "Cannot open video, uri or surface texture is null.");
        return;
    }
}

```

```

// 告诉音乐播放服务暂停// TODO: 这些常量需要在框架的某处发布。
Intent i = new Intent("com.android.music.musicservicecommand");
i.putExtra("command", "pause");
mContext.sendBroadcast(i);
release(false);
try {
    mSurface = new Surface(mSurfaceTexture);
    mMediaPlayer = new MediaPlayer();
    if (mAudioSession != 0) {
        mMediaPlayer.setAudioSessionId(mAudioSession);
    } else {
        mAudioSession = mMediaPlayer.getAudioSessionId();
    }
}

mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
    mMediaPlayer.setOnCompletionListener(mCompleteListener);
mMediaPlayer.setOnPreparedListener(mPreparedListener);
    mMediaPlayer.setOnErrorListener(mErrorListener);
mMediaPlayer.setOnInfoListener(mOnInfoListener);
    mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

    mMediaPlayer.setSurface(mSurface);
mCurrentBufferPercentage = 0;
    mMediaPlayer.setDataSource(mContext, mUri);

    mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
    mMediaPlayer.setScreenOnWhilePlaying(true);

mMediaPlayer.prepareAsync();
    mCurrentState = STATE_PREPARING;
} catch (IllegalStateException e) {
    mCurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO 自动生成的 catch 块
} catch (IOException e) {
mCurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO 自动生成的 catch 块
}
}

public void stopPlayback() {
    if (mMediaPlayer != null) {
        mMediaPlayer.stop();
        mMediaPlayer.release();
        mMediaPlayer = null;
        if (null != mMediaControllListener) {
            mMediaControllListener.onStop();
        }
    }
}

public void setMediaController(MediaController controller) {
    if (mMediaController != null) {
        mMediaController.hide();
    }
    mMediaController = controller;
    attachMediaController();
}

```

```

// Tell the music playback service to pause
// TODO: these constants need to be published somewhere in the
// framework.
Intent i = new Intent("com.android.music.musicservicecommand");
i.putExtra("command", "pause");
mContext.sendBroadcast(i);
release(false);
try {
    mSurface = new Surface(mSurfaceTexture);
    mMediaPlayer = new MediaPlayer();
    if (mAudioSession != 0) {
        mMediaPlayer.setAudioSessionId(mAudioSession);
    } else {
        mAudioSession = mMediaPlayer.getAudioSessionId();
    }
}

mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
mMediaPlayer.setOnCompletionListener(mCompleteListener);
mMediaPlayer.setOnPreparedListener(mPreparedListener);
mMediaPlayer.setOnErrorListener(mErrorListener);
mMediaPlayer.setOnInfoListener(mOnInfoListener);
mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

    mMediaPlayer.setSurface(mSurface);
mCurrentBufferPercentage = 0;
    mMediaPlayer.setDataSource(mContext, mUri);

    mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
    mMediaPlayer.setScreenOnWhilePlaying(true);

mMediaPlayer.prepareAsync();
    mCurrentState = STATE_PREPARING;
} catch (IllegalStateException e) {
    mCurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO auto-generated catch block
} catch (IOException e) {
    mCurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO auto-generated catch block
}
}

public void stopPlayback() {
    if (mMediaPlayer != null) {
        mMediaPlayer.stop();
        mMediaPlayer.release();
        mMediaPlayer = null;
        if (null != mMediaControllListener) {
            mMediaControllListener.onStop();
        }
    }
}

public void setMediaController(MediaController controller) {
    if (mMediaController != null) {
        mMediaController.hide();
    }
    mMediaController = controller;
    attachMediaController();
}

```

```

}

private void attachMediaController() {
    if (mMediaPlayer != null && mMediaController != null) {
        mMediaController.setMediaPlayer(this);
        View anchorView = this.getParent() instanceof View ? (View) this.getParent() : this;
        mMediaController.setAnchorView(anchorView);
        mMediaController.setEnabled(isInPlaybackState());
    }
}

private void release(boolean cleartargetstate) {
    Log.d(TAG, "释放媒体播放器。");
    if (mMediaPlayer != null) {
        mMediaPlayer.reset();
        mMediaPlayer.release();
        mMediaPlayer = null;
        mCurrentState = STATE_IDLE;
        if (cleartargetstate) {
            mTargetState = STATE_IDLE;
        }
    } 否则 {
        Log.d(TAG, "媒体播放器为空，未释放。");
    }
}

@Override
protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {
    // 如果已获取视频尺寸，将调整视图大小。
    // 视频尺寸是在 MediaPlayer 调用 onPrepared 后获取的
    //
    int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
    int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
    if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
        if ((mVideoWidth * height) > (width * mVideoHeight)) {
            Log.d(TAG, "视频太高，调整大小。");
            height = (width * mVideoHeight) / mVideoWidth;
        } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
            Log.d(TAG, "视频太宽，调整大小。");
            width = (height * mVideoWidth) / mVideoHeight;
        } else {
            Log.d(TAG, "宽高比正确。");
        }
    }
    setMeasuredDimension(width, height);
}

@Override
public boolean onTouchEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}

@Override
public boolean onTrackballEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}

```

```

}

private void attachMediaController() {
    if (mMediaPlayer != null && mMediaController != null) {
        mMediaController.setMediaPlayer(this);
        View anchorView = this.getParent() instanceof View ? (View) this.getParent() : this;
        mMediaController.setAnchorView(anchorView);
        mMediaController.setEnabled(isInPlaybackState());
    }
}

private void release(boolean cleartargetstate) {
    Log.d(TAG, "Releasing media player.");
    if (mMediaPlayer != null) {
        mMediaPlayer.reset();
        mMediaPlayer.release();
        mMediaPlayer = null;
        mCurrentState = STATE_IDLE;
        if (cleartargetstate) {
            mTargetState = STATE_IDLE;
        }
    } else {
        Log.d(TAG, "Media player was null, did not release.");
    }
}

@Override
protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {
    // Will resize the view if the video dimensions have been found.
    // video dimensions are found after onPrepared has been called by
    // MediaPlayer
    int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
    int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
    if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
        if ((mVideoWidth * height) > (width * mVideoHeight)) {
            Log.d(TAG, "Video too tall, change size.");
            height = (width * mVideoHeight) / mVideoWidth;
        } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
            Log.d(TAG, "Video too wide, change size.");
            width = (height * mVideoWidth) / mVideoHeight;
        } else {
            Log.d(TAG, "Aspect ratio is correct.");
        }
    }
    setMeasuredDimension(width, height);
}

@Override
public boolean onTouchEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}

@Override
public boolean onTrackballEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}

```

```

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode !=
KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
        && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU &&
keyCode != KeyEvent.KEYCODE_CALL
        && keyCode != KeyEvent.KEYCODE_ENDCALL;
    if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
        if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode ==
KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
                pause();
                mMediaController.show();
            } else {
start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
            if (!mMediaPlayer.isPlaying()) {
start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode ==
KeyEvent.KEYCODE_MEDIA_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
pause();
mMediaController.show();
            }
            return true;
        } 否则 {
toggleMediaControlsVisibility();
        }
    }
    return super.onKeyDown(keyCode, event);
}

private void toggleMediaControlsVisibility() {
    if (mMediaController.isShowing()) {
mMediaController.hide();
    } else {
mMediaController.show();
    }
}

public void start() {
    // 这可能会在多个点被调用，它会在所有条件准备好时执行// 1. 设置视频URI时// 2. 当Surf
ace可用时// 3. 来自Activity时
    // 1. 设置视频URI时
    // 2. 当Surface可用时
    // 3. 来自Activity时
    if (isInPlaybackState()) {
        mMediaPlayer.start();
        mCurrentState = STATE_PLAYING;
        if (null != mMediaControlListener) {
            mMediaControlListener.onStart();
        }
    } 否则 {
Log.d(TAG, "无法开始。当前状态 " + mCurrentState);
    }
}

```

```

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode !=
KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
        && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU &&
keyCode != KeyEvent.KEYCODE_CALL
        && keyCode != KeyEvent.KEYCODE_ENDCALL;
    if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
        if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode ==
KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
                pause();
                mMediaController.show();
            } else {
                start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
            if (!mMediaPlayer.isPlaying()) {
                start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode ==
KeyEvent.KEYCODE_MEDIA_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
                pause();
                mMediaController.show();
            }
            return true;
        } else {
            toggleMediaControlsVisibility();
        }
    }
    return super.onKeyDown(keyCode, event);
}

private void toggleMediaControlsVisibility() {
    if (mMediaController.isShowing()) {
        mMediaController.hide();
    } else {
        mMediaController.show();
    }
}

public void start() {
    // This can potentially be called at several points, it will go through
    // when all conditions are ready
    // 1. When setting the video URI
    // 2. When the surface becomes available
    // 3. From the activity
    if (isInPlaybackState()) {
        mMediaPlayer.start();
        mCurrentState = STATE_PLAYING;
        if (null != mMediaControlListener) {
            mMediaControlListener.onStart();
        }
    } else {
        Log.d(TAG, "Could not start. Current state " + mCurrentState);
    }
}

```

```

mTargetState = STATE_PLAYING;
}

public void pause() {
    if (isInPlaybackState()) {
        if (mMediaPlayer.isPlaying()) {
            mMediaPlayer.pause();
            mCurrentState = STATE_PAUSED;
            if (null != mMediControllListener) {
                mMediControllListener.onPause();
            }
        }
    }
    mTargetState = STATE_PAUSED;
}

public void suspend() {
    release(false);
}

public void resume() {
    openVideo();
}

@Override
public int getDuration() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getDuration();
    }

    return -1;
}

@Override
public int getCurrentPosition() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getCurrentPosition();
    }
    return 0;
}

@Override
public void seekTo(int msec) {
    if (isInPlaybackState()) {
        mMediaPlayer.seekTo(msec);
        mSeekWhenPrepared = 0;
    } else {
        mSeekWhenPrepared = msec;
    }
}

@Override
public boolean isPlaying() {
    return isInPlaybackState() && mMediaPlayer.isPlaying();
}

@Override
public int getBufferPercentage() {
    if (mMediaPlayer != null) {
        return mCurrentBufferPercentage;
    }
    return 0;
}

```

```

mTargetState = STATE_PLAYING;
}

public void pause() {
    if (isInPlaybackState()) {
        if (mMediaPlayer.isPlaying()) {
            mMediaPlayer.pause();
            mCurrentState = STATE_PAUSED;
            if (null != mMediControllListener) {
                mMediControllListener.onPause();
            }
        }
    }
    mTargetState = STATE_PAUSED;
}

public void suspend() {
    release(false);
}

public void resume() {
    openVideo();
}

@Override
public int getDuration() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getDuration();
    }

    return -1;
}

@Override
public int getCurrentPosition() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getCurrentPosition();
    }
    return 0;
}

@Override
public void seekTo(int msec) {
    if (isInPlaybackState()) {
        mMediaPlayer.seekTo(msec);
        mSeekWhenPrepared = 0;
    } else {
        mSeekWhenPrepared = msec;
    }
}

@Override
public boolean isPlaying() {
    return isInPlaybackState() && mMediaPlayer.isPlaying();
}

@Override
public int getBufferPercentage() {
    if (mMediaPlayer != null) {
        return mCurrentBufferPercentage;
    }
    return 0;
}

```

```

}

private boolean isInPlaybackState() {
    return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) && (mCurrentState != STATE_IDLE) && (mCurrentState != STATE_PREPARING));
}

@Override
public boolean canPause() {
    return false;
}

@Override
public boolean canSeekBackward() {
    return false;
}

@Override
public boolean canSeekForward() {
    return false;
}

@Override
public int getAudioSessionId() {
    if (mAudioSession == 0) {
        MediaPlayer foo = new MediaPlayer();
        mAudioSession = foo.getAudioSessionId();
        foo.release();
    }
    return mAudioSession;
}

// 监听器
private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
    @Override
    public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
        mCurrentBufferPercentage = percent;
    }
};

private MediaPlayer.OnCompletionListener mCompleteListener = new
MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(final MediaPlayer mp) {
        mCurrentState = STATE_PLAYBACK_COMPLETED;
        mTargetState = STATE_PLAYBACK_COMPLETED;
        mSurface.release();

        if (mMediaController != null) {
            mMediaController.hide();
        }

        if (mOnCompletionListener != null) {
            mOnCompletionListener.onCompletion(mp);
        }

        if (mMediaControllListener != null) {
            mMediaControllListener.onComplete();
        }
    }
};

```

```

}

private boolean isInPlaybackState() {
    return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) && (mCurrentState != STATE_IDLE) && (mCurrentState != STATE_PREPARING));
}

@Override
public boolean canPause() {
    return false;
}

@Override
public boolean canSeekBackward() {
    return false;
}

@Override
public boolean canSeekForward() {
    return false;
}

@Override
public int getAudioSessionId() {
    if (mAudioSession == 0) {
        MediaPlayer foo = new MediaPlayer();
        mAudioSession = foo.getAudioSessionId();
        foo.release();
    }
    return mAudioSession;
}

// Listeners
private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
    @Override
    public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
        mCurrentBufferPercentage = percent;
    }
};

private MediaPlayer.OnCompletionListener mCompleteListener = new
MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(final MediaPlayer mp) {
        mCurrentState = STATE_PLAYBACK_COMPLETED;
        mTargetState = STATE_PLAYBACK_COMPLETED;
        mSurface.release();

        if (mMediaController != null) {
            mMediaController.hide();
        }

        if (mOnCompletionListener != null) {
            mOnCompletionListener.onCompletion(mp);
        }

        if (mMediaControllListener != null) {
            mMediaControllListener.onComplete();
        }
    }
};

```

```

private MediaPlayer.OnPreparedListener mPreparedListener = new MediaPlayer.OnPreparedListener()
{
    @Override
    public void onPrepared(final MediaPlayer mp) {
        mCurrentState = STATE_PREPARED;

        mMediaController = new MediaController(getContext());

        if (mOnPreparedListener != null) {
            mOnPreparedListener.onPrepared(mMediaPlayer);
        }
        if (mMediaController != null) {
            mMediaController.setEnabled(true);
            //mMediaController.setAnchorView(getRootView());
        }
    }

    mVideoWidth = mp.getVideoWidth();
    mVideoHeight = mp.getVideoHeight();

    int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared may be
    // changed after seekTo()
    // call
    if (seekToPosition != 0) {
        seekTo(seekToPosition);
    }

    requestLayout();
    invalidate();
    if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
        if (mTargetState == STATE_PLAYING) {
            mMediaPlayer.start();
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        }
    } 否则 {
        if (mTargetState == STATE_PLAYING) {
            mMediaPlayer.start();
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        }
    }
}

private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
    @Override
    public void onVideoSizeChanged(final MediaPlayer mp, final int width, final int height) {
        mVideoWidth = mp.getVideoWidth();
        mVideoHeight = mp.getVideoHeight();
        if (mVideoWidth != 0 && mVideoHeight != 0) {
            requestLayout();
        }
    }
};

private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(final MediaPlayer mp, final int what, final int extra) {
        Log.d(TAG, "Error: " + what + "," + extra);
    }
};

```

```

private MediaPlayer.OnPreparedListener mPreparedListener = new MediaPlayer.OnPreparedListener()
{
    @Override
    public void onPrepared(final MediaPlayer mp) {
        mCurrentState = STATE_PREPARED;

        mMediaController = new MediaController(getContext());

        if (mOnPreparedListener != null) {
            mOnPreparedListener.onPrepared(mMediaPlayer);
        }
        if (mMediaController != null) {
            mMediaController.setEnabled(true);
            //mMediaController.setAnchorView(getRootView());
        }
    }

    mVideoWidth = mp.getVideoWidth();
    mVideoHeight = mp.getVideoHeight();

    int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared may be
    // changed after seekTo()
    // call
    if (seekToPosition != 0) {
        seekTo(seekToPosition);
    }

    requestLayout();
    invalidate();
    if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
        if (mTargetState == STATE_PLAYING) {
            mMediaPlayer.start();
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        }
    } else {
        if (mTargetState == STATE_PLAYING) {
            mMediaPlayer.start();
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        }
    }
};

private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
    @Override
    public void onVideoSizeChanged(final MediaPlayer mp, final int width, final int height) {
        mVideoWidth = mp.getVideoWidth();
        mVideoHeight = mp.getVideoHeight();
        if (mVideoWidth != 0 && mVideoHeight != 0) {
            requestLayout();
        }
    }
};

private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(final MediaPlayer mp, final int what, final int extra) {
        Log.d(TAG, "Error: " + what + "," + extra);
    }
};

```

```

mCurrentState = STATE_ERROR;
mTargetState = STATE_ERROR;

if (mMediaController != null) {
    mMediaController.hide();
}

/* 如果提供了错误处理程序，则使用它并结束。*/
if (mOnErrorHandler != null) {
    if (mOnErrorHandler.onError(mMediaPlayer, what, extra)) {
        return true;
    }
}

/*
* 否则，弹出错误对话框，让用户知道发生了错误。只有在 * 我们附加到窗口时
才尝试弹出对话框。当我们即将关闭且不再 * 拥有窗口时，不必向用户显示错误。
*/
if (getWindowToken() != null) {

//      new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," +
extra).setPositiveButton("OK", new DialogInterface.OnClickListener() {
//          public void onClick(DialogInterface dialog, int whichButton) {
//              /*
//                  * 如果执行到这里，说明没有 onError 监听器，至少
//                  * 通知他们视频已结束。
//              */
//              if (mOnCompletionListener != null) {
//                  mOnCompletionListener.onCompletion(mMediaPlayer);
//              }
//          }
//      }).setCancelable(false).show();
}
return true;
};


```

```

SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
    @Override
    public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width, final
int height) {
Log.d(TAG, "onSurfaceTextureAvailable.");
    mSurfaceTexture = surface;
openVideo();
}

    @Override
    public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width,
final int height) {
Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
    mSurfaceWidth = width;
mSurfaceHeight = height;
    boolean isValidState = (mTargetState == STATE_PLAYING);
    boolean hasValidSize = (mVideoWidth == width && mVideoHeight == height);
    if (mMediaPlayer != null && isValidState && hasValidSize) {
        if (mSeekWhenPrepared != 0) {
            seekTo(mSeekWhenPrepared);
        }
start();
}

```

```

mCurrentState = STATE_ERROR;
mTargetState = STATE_ERROR;

if (mMediaController != null) {
    mMediaController.hide();
}

/* If an error handler has been supplied, use it and finish. */
if (mOnErrorHandler != null) {
    if (mOnErrorHandler.onError(mMediaPlayer, what, extra)) {
        return true;
    }
}

/*
* Otherwise, pop up an error dialog so the user knows that
* something bad has happened. Only try and pop up the dialog if
* we're attached to a window. When we're going away and no longer
* have a window, don't bother showing the user an error.
*/
if (getWindowToken() != null) {

//      new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," +
extra).setPositiveButton("OK", new DialogInterface.OnClickListener() {
//          public void onClick(DialogInterface dialog, int whichButton) {
//              /*
//                  * If we get here, there is no onError listener, so at
//                  * least inform them that the video is over.
//              */
//              if (mOnCompletionListener != null) {
//                  mOnCompletionListener.onCompletion(mMediaPlayer);
//              }
//          }
//      }).setCancelable(false).show();
}
return true;
};


```

```

SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
    @Override
    public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width, final
int height) {
Log.d(TAG, "onSurfaceTextureAvailable.");
    mSurfaceTexture = surface;
openVideo();
}

    @Override
    public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width,
final int height) {
Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
    mSurfaceWidth = width;
mSurfaceHeight = height;
    boolean isValidState = (mTargetState == STATE_PLAYING);
    boolean hasValidSize = (mVideoWidth == width && mVideoHeight == height);
    if (mMediaPlayer != null && isValidState && hasValidSize) {
        if (mSeekWhenPrepared != 0) {
            seekTo(mSeekWhenPrepared);
        }
start();
}

```

```

    }

    @Override
    public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {
        mSurface = null;
        if (mMediaController != null)
            mMediaController.hide();
        release(true);
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(final SurfaceTexture surface) {
    }

};

/***
 * 注册一个回调，当媒体文件加载完成并准备好播放时调用。
 *
 * @param l 将要运行的回调
 */
public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
    mOnPreparedListener = l;
}

/***
 * 注册一个回调，当播放过程中达到媒体文件末尾时调用。
 *
 * @param l 将要运行的回调
 */
public void setOnCompletionListener(OnCompletionListener l) {
    mOnCompletionListener = l;
}

/***
 * 注册一个回调函数，当播放或设置过程中发生错误时调用。
 * 如果未指定监听器，或者监听器返回 false,
 * VideoView 将通知用户任何错误。
 *
 * @param l 将要运行的回调
 */
public void setOnErrorListener(OnErrorListener l) {
    mOnErrorListener = l;
}

/***
 * 注册一个回调，当播放或设置过程中发生信息事件时调用。
 *
 * @param l 将要运行的回调
 */
public void setOnInfoListener(OnInfoListener l) {
    mOnInfoListener = l;
}

public static interface MediaControllerListener {
    public void onStart();
}

```

```

    }

    @Override
    public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {
        mSurface = null;
        if (mMediaController != null)
            mMediaController.hide();
        release(true);
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(final SurfaceTexture surface) {
    }

    /**
     * Register a callback to be invoked when the media file is loaded and ready
     * to go.
     *
     * @param l The callback that will be run
     */
    public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
        mOnPreparedListener = l;
    }

    /**
     * Register a callback to be invoked when the end of a media file has been
     * reached during playback.
     *
     * @param l The callback that will be run
     */
    public void setOnCompletionListener(OnCompletionListener l) {
        mOnCompletionListener = l;
    }

    /**
     * Register a callback to be invoked when an error occurs during playback or
     * setup. If no listener is specified, or if the listener returned false,
     * VideoView will inform the user of any errors.
     *
     * @param l The callback that will be run
     */
    public void setOnErrorListener(OnErrorListener l) {
        mOnErrorListener = l;
    }

    /**
     * Register a callback to be invoked when an informational event occurs
     * during playback or setup.
     *
     * @param l The callback that will be run
     */
    public void setOnInfoListener(OnInfoListener l) {
        mOnInfoListener = l;
    }

    public static interface MediaControllerListener {
        public void onStart();
    }

```

```
public void onPause();
public void onStop();
public void onComplete();
}

MediaControllListener mMediaControllListener;

public void setMediaControllListener(MediaControllListener mediaControllListener) {
    mMediaControllListener = mediaControllListener;
}

@Override
public void setVisibility(int visibility) {
    System.out.println("setVisibility: " + visibility);
    super.setVisibility(visibility);
}
}
```

来自这个gitub 仓库的帮助。虽然它有一些问题，因为它是三年前写的，但正如上面所述，我自己设法修复了它们。

```
public void onPause();
public void onStop();
public void onComplete();
}

MediaControllListener mMediaControllListener;

public void setMediaControllListener(MediaControllListener mediaControllListener) {
    mMediaControllListener = mediaControllListener;
}

@Override
public void setVisibility(int visibility) {
    System.out.println("setVisibility: " + visibility);
    super.setVisibility(visibility);
}
}
```

Help from this [gitub repository](#). Though It has some issues as it was written 3 years ago I managed to fix them on my own as written above.

第24章：WebView

[WebView](#) 是一个在应用程序内显示网页的视图。通过它你可以添加你自己的URL。

第24.1节：通过打印控制台消息或远程调试排查WebView问题

将WebView控制台消息打印到logcat

要处理网页的控制台消息，你可以重写WebChromeClient中的onConsoleMessage方法：

```
final class ChromeClient extends WebChromeClient {  
    @Override  
    public boolean onConsoleMessage(ConsoleMessage msg) {  
        Log.d(  
            "WebView",  
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())  
        );  
        return true;  
    }  
}
```

然后在你的活动或片段中设置它：

```
webView.setWebChromeClient(new ChromeClient());
```

所以这个示例页面：

```
<html>  
<head>  
    <script type="text/javascript">  
        console.log('test message');  
    </script>  
</head>  
<body>  
</body>  
</html>
```

将会向logcat写入日志 'test message'：

```
WebView: test message sample.html:4
```

console.info()、console.warn() 和 console.error() 也被chrome-client支持。

使用Chrome远程调试安卓设备

您可以通过桌面版Chrome远程调试基于WebView的应用程序。

在您的安卓设备上启用USB调试

在您的安卓设备上，打开设置，找到开发者选项部分，启用USB调试。

连接并发现您的安卓设备

Chapter 24: WebView

[WebView](#) is a view that display web pages inside your application. By this you can add your own URL.

Section 24.1: Troubleshooting WebView by printing console messages or by remote debugging

Printing webview console messages to logcat

To handle console messages from web page you can override [onConsoleMessage](#) in [WebChromeClient](#):

```
final class ChromeClient extends WebChromeClient {  
    @Override  
    public boolean onConsoleMessage(ConsoleMessage msg) {  
        Log.d(  
            "WebView",  
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())  
        );  
        return true;  
    }  
}
```

And set it in your activity or fragment:

```
webView.setWebChromeClient(new ChromeClient());
```

So this sample page:

```
<html>  
<head>  
    <script type="text/javascript">  
        console.log('test message');  
    </script>  
</head>  
<body>  
</body>  
</html>
```

will write log 'test message' to logcat:

```
WebView: test message sample.html:4
```

console.info()，console.warn() 和 console.error() 也支持。

Remote debugging android devices with Chrome

您可以从桌面版Chrome远程调试基于WebView的应用程序。

Enable USB debugging on your Android device

在您的Android设备上，打开设置，找到Developer options部分，启用USB调试。

Connect and discover your Android device

在Chrome中打开以下页面：chrome://inspect/#devices在“检查设备”对话框中，选择您的设备并按inspect。Chrome的DevTools新实例将在您的开发机器上打开。

有关DevTools的更详细指南和说明可在developers.google.com找到

第24.2节：从Javascript到Java的通信 (安卓)

安卓活动

```
包com.example.myapp;

导入android.os.Bundle;
导入android.app.Activity;
导入android.webkit.WebView;

公共类WebViewActivity extends Activity {
    @Override
    保护 void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        setContentView(webView);

        /*
        * 注意标签Android, 这在Javascript端使用
        * 当然, 您可以更改它。
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript 处理器

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     * 关键是注解 @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
        // 执行某些操作
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
        // 执行某些操作
    }
}
```

网页，Javascript 调用

Open page in chrome following page: <chrome://inspect/#devices>

From the Inspect Devices dialog, select your device and press **inspect**. A new instance of Chrome's DevTools opens up on your development machine.

More detailed guideline and description of DevTools can be found on developers.google.com

Section 24.2: Communication from Javascript to Java (Android)

Android Activity

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
    保护 void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        setContentView(webView);

        /*
        * Note the label Android, this is used in the Javascript side of things
        * You can of course change this.
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript Handler

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     * Key point here is the annotation @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
        // Do something
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
        // Do something
    }
}
```

Web Page, Javascript call

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

额外提示

传入复杂数据结构的一个可能解决方案是使用 JSON。

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

在 Android 端使用你喜欢的 JSON 解析器，例如：JSONObject

第24.3节：从 Java 到 Javascript 的通信

基本示例

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        webView = new WebView(this);
        webView.getSettings().setJavaScriptEnabled(true);

        setContentView(webView);

        webView.loadUrl("http://example.com");

        /*
        * 调用 Javascript 函数
        */
        webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

    /**
     * 调用 Javascript 函数
     */
    public void doSomething() {
        this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
    }
}
```

第24.4节：打开拨号器示例

如果网页中包含电话号码，你可以使用手机的拨号器拨打电话。此代码检查

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Extra Tip

Passing in a complex data structure, a possible solution is use JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

On the Android side use your favorite JSON parser ie: JSONObject

Section 24.3: Communication from Java to Javascript

Basic Example

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        webView = new WebView(this);
        webView.getSettings().setJavaScriptEnabled(true);

        setContentView(webView);

        webView.loadUrl("http://example.com");

        /*
        * Invoke Javascript function
        */
        webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

    /**
     * Invoking a Javascript function
     */
    public void doSomething() {
        this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
    }
}
```

Section 24.4: Open dialer example

If the web page contains phone number you can make a call using your phone's dialer. This code checks for the

以 tel: 开头的 URL，然后创建一个意图打开拨号器，你可以拨打点击的电话号码：

```
public boolean shouldOverrideUrlLoading(WebView 视图, String url) {  
    if (url.startsWith("tel:")) {  
        Intent 意图 = new Intent(Intent.ACTION_DIAL,  
            Uri.parse(url));  
        startActivity(意图);  
    } else if(url.startsWith("http:") || url.startsWith("https:")) {  
        视图.loadUrl(url);  
    }  
    return true;  
}
```

第24.5节：在WebView中打开本地文件 / 创建动态内容

布局.xml

```
<WebView  
    android:id="@+id/WebViewToDisplay"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_gravity="center"  
    android:fadeScrollbars="false" />
```

加载数据到 WebViewToDisplay

```
WebView webViewDisplay;  
StringBuffer LoadWEb1;  
  
webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);  
LoadWEb1 = new StringBuffer();  
LoadWEb1.append("<html><body><h1>我的第一个标题</h1><p>我的第一段。</p>");  
//运行时读取参数的示例代码  
String strName = "测试段落";  
LoadWEb1.append("<br/><p>" + strName + "</p>");  
String result = LoadWEb1.append("</body></html>").toString();  
WebSettings webSettings = webViewDisplay.getSettings();  
webSettings.setJavaScriptEnabled(true);  
webViewDisplay.getSettings().setBuiltInZoomControls(true);  
if (android.os.Build.VERSION.SDK_INT >= 11){  
    webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);  
    webViewDisplay.getSettings().setDisplayZoomControls(false);  
}  
  
webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8",  
        null);  
//要直接从assets文件夹加载本地文件，请使用以下代码  
//webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

第24.6节：WebView中的JavaScript警告对话框 - 如何使其正常工作

默认情况下，WebView 不会实现 JavaScript 警告对话框，即 alert() 不会有任何反应。为了使其生效，您首先需要启用 JavaScript（显然..），然后设置一个 WebChromeClient 来处理页面发出的警告对话框请求：

url which starts with tel: then make an intent to open dialer and you can make a call to the clicked phone number:

```
public boolean shouldOverrideUrlLoading(WebView view, String url) {  
    if (url.startsWith("tel:")) {  
        Intent intent = new Intent(Intent.ACTION_DIAL,  
            Uri.parse(url));  
        startActivity(intent);  
    } else if(url.startsWith("http:") || url.startsWith("https:")) {  
        view.loadUrl(url);  
    }  
    return true;  
}
```

Section 24.5: Open Local File / Create dynamic content in Webview

Layout.xml

```
<WebView  
    android:id="@+id/WebViewToDisplay"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_gravity="center"  
    android:fadeScrollbars="false" />
```

Load data into WebViewToDisplay

```
WebView webViewDisplay;  
StringBuffer LoadWEb1;  
  
webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);  
LoadWEb1 = new StringBuffer();  
LoadWEb1.append("<html><body><h1>My First Heading</h1><p>My first paragraph.</p>");  
//Sample code to read parameters at run time  
String strName = "Test Paragraph";  
LoadWEb1.append("<br/><p>" + strName + "</p>");  
String result = LoadWEb1.append("</body></html>").toString();  
WebSettings webSettings = webViewDisplay.getSettings();  
webSettings.setJavaScriptEnabled(true);  
webViewDisplay.getSettings().setBuiltInZoomControls(true);  
if (android.os.Build.VERSION.SDK_INT >= 11){  
    webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);  
    webViewDisplay.getSettings().setDisplayZoomControls(false);  
}  
  
webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8",  
        null);  
//To load local file directly from assets folder use below code  
//webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

Section 24.6: JavaScript alert dialogs in WebView - How to make them work

By default, WebView does not implement JavaScript alert dialogs, ie. alert() will do nothing. In order to make you need to firstly enable JavaScript (obviously..), and then set a WebChromeClient to handle requests for alert dialogs from the page:

```
webView.setWebChromeClient(new WebChromeClient() {  
    //如果需要，这里可以为您的 WebChromeClient 添加其他方法..  
  
    @Override  
    public boolean onJsAlert(WebView view, String url, String message, JsResult result) {  
        return super.onJsAlert(view, url, message, result);  
    }  
});
```

这里，我们重写了 `onJsAlert`，然后调用了父类的实现，这会给我们一个标准的 Android 对话框。您也可以自行使用 `message` 和 `URL`，例如如果您想创建一个自定义样式的对话框，或者想要记录它们。

```
webView.setWebChromeClient(new WebChromeClient() {  
    //Other methods for your WebChromeClient here, if needed..  
  
    @Override  
    public boolean onJsAlert(WebView view, String url, String message, JsResult result) {  
        return super.onJsAlert(view, url, message, result);  
    }  
});
```

Here, we override `onJsAlert`, and then we call through to the super implementation, which gives us a standard Android dialog. You can also use the message and URL yourself, for example if you want to create a custom styled dialog or if you want to log them.

第25章：SearchView

第25.1节：为 SearchView 设置主题

基本上，要为从 menu.xml 中提取的 app:actionViewClass 的 SearchView 应用主题，我们需要理解这完全取决于应用于底层工具栏（Toolbar）的样式。要实现工具栏的主题化，请执行以下步骤。

在 styles.xml 中创建一个样式

```
<style name="ActionBarThemeOverlay">
    <item name="android:textColorPrimary">@color/prim_color</item>
    <item name="colorControlNormal">@color/normal_color</item>
    <item name="colorControlHighlight">@color/high_color</item>
    <item name="android:textColorHint">@color/hint_color</item>
</style>
```

将样式应用到工具栏。

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    app:theme="@style/ActionBarThemeOverlay"
    app:popupTheme="@style/ActionBarThemeOverlay"
    android:layout_width="match_parent"
    android:layout_height="?attr actionBarSize"
    android:background="@color/colorPrimary"
    android:title="@string/title"
    tools:targetApi="m" />
```

这会为与工具栏对应的所有视图（返回按钮、菜单图标和搜索视图）赋予所需的颜色。

第25.2节：带有Fragment的工具栏中的搜索视图

menu.xml - (res -> menu)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".HomeActivity">

    <item
        android:id="@+id/action_search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="搜索"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always" />

</menu>
```

MainFragment.java

```
public class MainFragment extends Fragment {

    private SearchView searchView = null;
    private SearchView.OnQueryTextListener queryTextListener;
```

Chapter 25: SearchView

Section 25.1: Setting Theme for SearchView

Basically to apply a theme for SearchView extracted as app:actionViewClass from the menu.xml, we need understand that it depends completely on the style applied to the underlying Toolbar. To achieve themeing the Toolbar apply the following steps.

Create a style in the styles.xml

```
<style name="ActionBarThemeOverlay">
    <item name="android:textColorPrimary">@color/prim_color</item>
    <item name="colorControlNormal">@color/normal_color</item>
    <item name="colorControlHighlight">@color/high_color</item>
    <item name="android:textColorHint">@color/hint_color</item>
</style>
```

Apply the style to the Toolbar.

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    app:theme="@style/ActionBarThemeOverlay"
    app:popupTheme="@style/ActionBarThemeOverlay"
    android:layout_width="match_parent"
    android:layout_height="?attr actionBarSize"
    android:background="@color/colorPrimary"
    android:title="@string/title"
    tools:targetApi="m" />
```

This gives the desired color to the all the views corresponding to the Toolbar (back button, Menu icons and SearchView).

Section 25.2: SearchView in Toolbar with Fragment

menu.xml - (res -> menu)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".HomeActivity">

    <item
        android:id="@+id/action_search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="Search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always" />

</menu>
```

MainFragment.java

```
public class MainFragment extends Fragment {

    private SearchView searchView = null;
    private SearchView.OnQueryTextListener queryTextListener;
```

```

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_main, container, false);
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setHasOptionsMenu(true);
}

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    inflater.inflate(R.menu.menu, menu);
    MenuItem searchItem = menu.findItem(R.id.action_search);
    SearchManager searchManager = (SearchManager)
getActivity().getSystemService(Context.SEARCH_SERVICE);

    if (searchItem != null) {
searchView = (SearchView) searchItem.getActionView();
    }
    if (searchView != null) {

searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));

queryTextListener = new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextChange(String newText) {
            Log.i("onQueryTextChange", newText);

            return true;
        }
        @Override
        public boolean onQueryTextSubmit(String query) {
            Log.i("onQueryTextSubmit", query);

            return true;
        }
    };
searchView.setOnQueryTextListener(queryTextListener);
    super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_search:
            // Not implemented here
            return false;
        default:
            break;
    }
searchView.setOnQueryTextListener(queryTextListener);
    return super.onOptionsItemSelected(item);
}

```

```

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_main, container, false);
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setHasOptionsMenu(true);
}

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    inflater.inflate(R.menu.menu, menu);
    MenuItem searchItem = menu.findItem(R.id.action_search);
    SearchManager searchManager = (SearchManager)
getActivity().getSystemService(Context.SEARCH_SERVICE);

    if (searchItem != null) {
searchView = (SearchView) searchItem.getActionView();
    }
    if (searchView != null) {

searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));

queryTextListener = new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextChange(String newText) {
            Log.i("onQueryTextChange", newText);

            return true;
        }
        @Override
        public boolean onQueryTextSubmit(String query) {
            Log.i("onQueryTextSubmit", query);

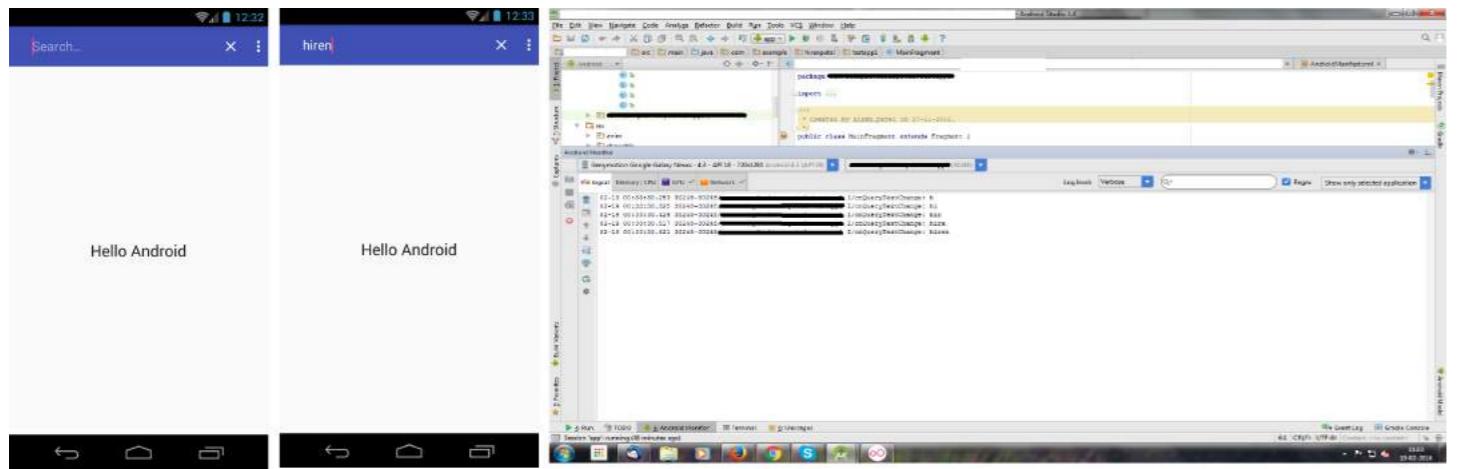
            return true;
        }
    };
searchView.setOnQueryTextListener(queryTextListener);
    super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_search:
            // Not implemented here
            return false;
        default:
            break;
    }
searchView.setOnQueryTextListener(queryTextListener);
    return super.onOptionsItemSelected(item);
}

```

参考截图：

Reference screenshot:



第25.3节：带有RxBindings监听器的Appcompat SearchView

build.gradle:

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu/menu.xml:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="搜索"
          android:icon="@android:drawable/ic_menu_search"
          app:actionViewClass="android.support.v7.widget.SearchView"
          app:showAsAction="always"/>
</menu>
```

MainActivity.java:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

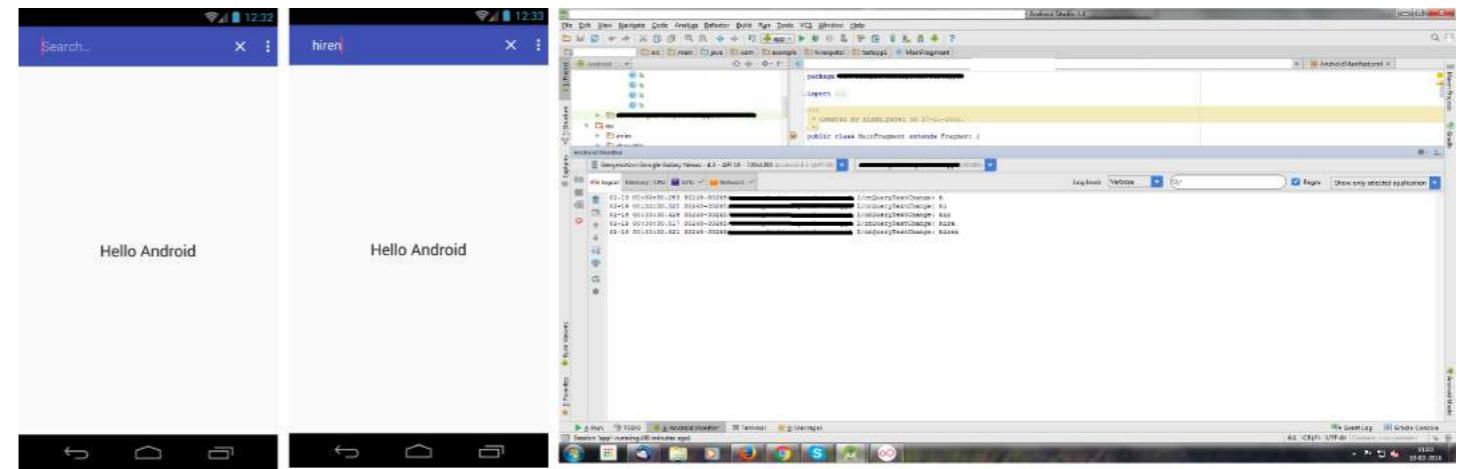
    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem);

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // 你的提示文字

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // 可选：设置开始搜索的字母数为1
    // 默认是2
    try {
        int autoCompleteTextViewID = getResources().getIdentifier("android:id/search_src_text",
            null, null);
    }
}
```



Section 25.3: Appcompat SearchView with RxBindings watcher

build.gradle:

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu/menu.xml:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
          android:icon="@android:drawable/ic_menu_search"
          app:actionViewClass="android.support.v7.widget.SearchView"
          app:showAsAction="always"/>
</menu>
```

MainActivity.java:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem);

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID = getResources().getIdentifier("android:id/search_src_text",
            null, null);
    }
}
```

```

AutoCompleteTextView search.AutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
search.AutoCompleteTextView.setThreshold(1);
} catch (Exception e) {
Logs.e(TAG, "设置搜索视图字母阈值失败");
}

searchView.setOnSearchClickListener(v -> {
    // 搜索视图展开时的可选操作
});
searchView.setOnCloseListener(() -> {
    // 可选操作：搜索视图关闭时
    return false;
});

RxSearchView.queryTextChanges(searchView)
    .doOnEach(notification -> {
CharSequence query = (CharSequence) notification.getValue();
    searchAdapter.filter(query);
})
.debounce(300, TimeUnit.MILLISECONDS) // 跳过中间字母
    .flatMap(query -> MyWebService.search(query)) // 发起搜索请求
    .retry(3)
.subscribe(results -> {
    searchAdapter.populateAdapter(results);
});

// 可选：关闭时折叠搜索视图
searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
    if (!queryTextFocused) {
        collapseSearchView();
    }
});

```

SearchAdapter.java

```

public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
c.addRow(new Object[]{i});
            }
        }
    }
}

```

```

AutoCompleteTextView search.AutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
search.AutoCompleteTextView.setThreshold(1);
} catch (Exception e) {
Logs.e(TAG, "failed to set search view letters threshold");
}

searchView.setOnSearchClickListener(v -> {
    // optional actions to search view expand
});
searchView.setOnCloseListener(() -> {
    // optional actions to search view close
    return false;
});

RxSearchView.queryTextChanges(searchView)
    .doOnEach(notification -> {
CharSequence query = (CharSequence) notification.getValue();
    searchAdapter.filter(query);
})
.debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
    .flatMap(query -> MyWebService.search(query)) // make a search request
    .retry(3)
    .subscribe(results -> {
        searchAdapter.populateAdapter(results);
    });

//optional: collapse the searchView on close
searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
    if (!queryTextFocused) {
        collapseSearchView();
    }
});

```

SearchAdapter.java

```

public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
    }
}

```

```

        }
changeCursor(c);
    notifyDataSetChanged();
}

@Override
public void bindView(View view, Context context, Cursor cursor) {
    ViewHolder holder = (ViewHolder) view.getTag();
    int position = cursor.getPosition();
    if (position < items.size()) {
        SearchResult result = items.get(position);
        // 在这里绑定你的视图
    }
}

@Override
public View newView(Context context, Cursor cursor, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

    View v = inflater.inflate(R.layout.search_list_item, parent, false);
    ViewHolder holder = new ViewHolder(v);

    v.setTag(holder);
    return v;
}

private static class ViewHolder {
    public final TextView text;

    public ViewHolder(View v) {
        this.text= (TextView) v.findViewById(R.id.text);
    }
}

```

```

        }
changeCursor(c);
    notifyDataSetChanged();
}

@Override
public void bindView(View view, Context context, Cursor cursor) {
    ViewHolder holder = (ViewHolder) view.getTag();
    int position = cursor.getPosition();
    if (position < items.size()) {
        SearchResult result = items.get(position);
        // bind your view here
    }
}

@Override
public View newView(Context context, Cursor cursor, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

    View v = inflater.inflate(R.layout.search_list_item, parent, false);
    ViewHolder holder = new ViewHolder(v);

    v.setTag(holder);
    return v;
}

private static class ViewHolder {
    public final TextView text;

    public ViewHolder(View v) {
        this.text= (TextView) v.findViewById(R.id.text);
    }
}

```

第26章：底部导航视图

底部导航视图已经在材质设计指南中存在了一段时间，但我们一直很难将其实现到我们的应用中。

一些应用程序构建了自己的解决方案，而其他应用则依赖第三方开源库来完成这项工作。

现在设计支持库增加了这个底部导航栏，让我们深入了解如何使用它吧！

第26.1节：基本实现

要添加BottomNavigationView，请按照以下步骤操作：

1. 在你的build.gradle中添加依赖项：

```
compile 'com.android.support:design:25.1.0'
```

2. 在你的布局中添加BottomNavigationView：

```
<android.support.design.widget.BottomNavigationView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/bottom_navigation"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:menu="@menu/bottom_navigation_menu"/>
```

3. 创建menu以填充视图：

```
<!-- res/menu/bottom_navigation_menu.xml -->  
  
<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:app="http://schemas.android.com/apk/res-auto">  
    <item  
        android:id="@+id/my_action1"  
        android:enabled="true"  
        android:icon="@drawable/my_drawable"  
        android:title="@string/text"  
        app:showAsAction="ifRoom" />  
    ...  
</menu>
```

4. 为点击事件附加监听器：

```
//获取视图  
BottomNavigationView bottomNavigationView = (BottomNavigationView)  
        findViewById(R.id.bottom_navigation);  
  
//附加监听器  
bottomNavigationView.setOnNavigationItemSelectedListener(  
    new BottomNavigationView.OnNavigationItemSelectedListener() {  
        @Override  
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
            switch (item.getItemId()) {  
                case R.id.my_action1:
```

Chapter 26: BottomNavigationView

The Bottom Navigation View has been in the material design [guidelines](#) for some time, but it hasn't been easy for us to implement it into our apps.

Some applications have built their own solutions, whilst others have relied on third-party open-source libraries to get the job done.

Now the design support library is seeing the addition of this bottom navigation bar, let's take a dive into how we can use it!

Section 26.1: Basic implementation

To add the BottomNavigationView follow these steps:

1. Add in your build.gradle the **dependency**:

```
compile 'com.android.support:design:25.1.0'
```

2. Add the BottomNavigationView in **your layout**:

```
<android.support.design.widget.BottomNavigationView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/bottom_navigation"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:menu="@menu/bottom_navigation_menu"/>
```

3. Create the **menu** to populate the view:

```
<!-- res/menu/bottom_navigation_menu.xml -->  
  
<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:app="http://schemas.android.com/apk/res-auto">  
    <item  
        android:id="@+id/my_action1"  
        android:enabled="true"  
        android:icon="@drawable/my_drawable"  
        android:title="@string/text"  
        app:showAsAction="ifRoom" />  
    ...  
</menu>
```

4. Attach a **listener** for the click events:

```
//Get the view  
BottomNavigationView bottomNavigationView = (BottomNavigationView)  
        findViewById(R.id.bottom_navigation);  
  
//Attach the listener  
bottomNavigationView.setOnNavigationItemSelectedListener(  
    new BottomNavigationView.OnNavigationItemSelectedListener() {  
        @Override  
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
            switch (item.getItemId()) {  
                case R.id.my_action1:
```

```

    //执行某些操作...
    break;

    ...
}

return true; //返回false会禁用导航栏动画
);

```

查看演示代码：[BottomNavigation-Demo](#)

第26.2节：BottomNavigationView的自定义

注意：我假设你已经了解如何使用BottomNavigationView。

本示例将讲解如何为BottomNavigationView添加选择器。这样你可以为图标和

文本在界面上设置状态。

创建可绘制资源bottom_navigation_view_selector.xml为

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>

```

并在布局文件中的BottomNavigationView中使用以下属性

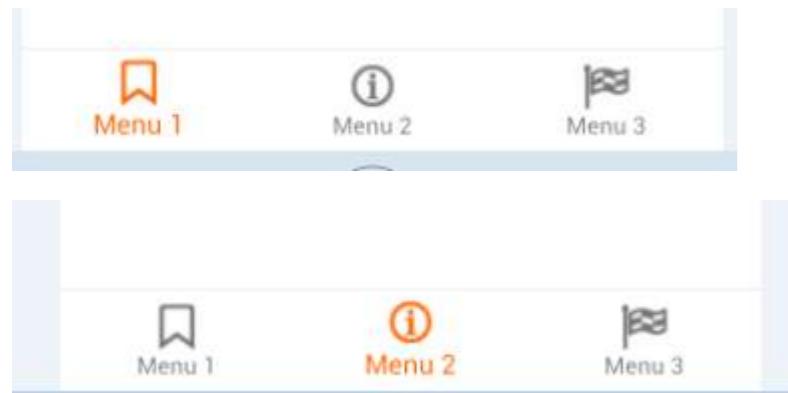
```

app:itemIconTint="@drawable/bottom_navigation_view_selector"
app:itemTextColor="@drawable/bottom_navigation_view_selector"

```

在上述示例中，我为app:itemIconTint和app:itemTextColor都使用了相同的选择器bottom_navigation_view_selector，以保持文本和图标颜色一致。但如果您的设计中文本和图标颜色不同，您可以定义两个不同的选择器并使用它们。

输出效果类似如下



第26.3节：处理启用/禁用状态

为启用/禁用菜单项创建选择器。

selector.xml

```

//Do something...
break;

...
}

return true; //returning false disables the Navigation bar animations
);

```

Checkout demo code at [BottomNavigation-Demo](#)

Section 26.2: Customization of BottomNavigationView

Note : I am assuming that you know about how to use BottomNavigationView.

This example I will explain how to add selector for BottomNavigationView. So you can state on UI for icons and texts.

Create drawable bottom_navigation_view_selector.xml as

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>

```

And use below attributes into BottomNavigationView in layout file

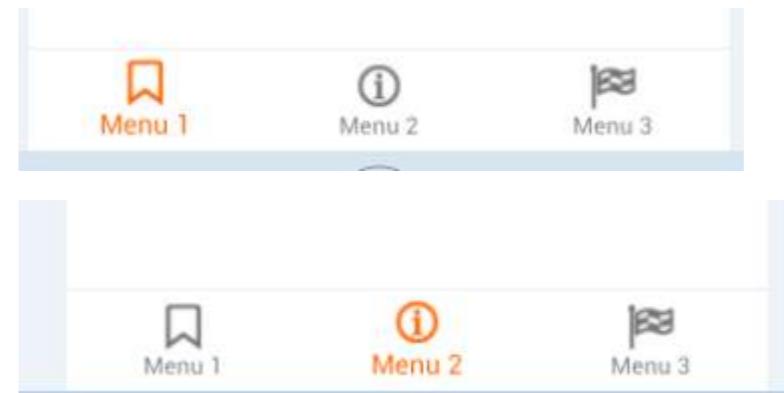
```

app:itemIconTint="@drawable/bottom_navigation_view_selector"
app:itemTextColor="@drawable/bottom_navigation_view_selector"

```

In above example, I have used same selector bottom_navigation_view_selector for app:itemIconTint and app:itemTextColor both to keep text and icon colors same. But if your design has different color for text and icon, you can define 2 different selectors and use them.

Output will be similar to below



Section 26.3: Handling Enabled / Disabled states

Create Selector for Enable/Disable Menu Item.

selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/white" android:state_enabled="true" />
    <item android:color="@color/colorPrimaryDark" android:state_enabled="false" />
</selector>
```

design.xml

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```

第26.4节：允许超过3个菜单

这个示例严格来说是一种变通方法，因为目前没有办法禁用一种称为ShiftMode的行为。

创建如下函数。

```
public static void disableMenuShiftMode(BottomNavigationView view) {
    BottomNavigationMenuView menuView = (BottomNavigationMenuView) view.getChildAt(0);
    try {
        Field shiftingMode = menuView.getClass().getDeclaredField("mShiftingMode");
        shiftingMode.setAccessible(true);
        shiftingMode.setBoolean(menuView, false);
        shiftingMode.setAccessible(false);
        for (int i = 0; i < menuView.getChildCount(); i++) {
            BottomNavigationItemView item = (BottomNavigationItemView) menuView.getChildAt(i);
            //noinspection RestrictedApi
            item.setShiftingMode(false);
            // 再次设置选中值，以便视图更新
            //noinspection RestrictedApi
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("BNVHelper", "无法获取 shift 模式字段", e);
    } catch (IllegalAccessException e) {
        Log.e("BNVHelper", "无法更改 shift 模式的值", e);
    }
}
```

当菜单项数量超过3个时，禁用菜单的切换 (Shifting) 行为。

用法

```
BottomNavigationView navView = (BottomNavigationView) findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Proguard 问题：还需在 Proguard 配置文件中添加以下行，否则此功能无法正常工作。

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
    boolean mShiftingMode;
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/white" android:state_enabled="true" />
    <item android:color="@color/colorPrimaryDark" android:state_enabled="false" />
</selector>
```

design.xml

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```

Section 26.4: Allowing more than 3 menus

This example is strictly a workaround since, currently there is no way to disable a behaviour known as ShiftMode.

Create a function as such.

```
public static void disableMenuShiftMode(BottomNavigationView view) {
    BottomNavigationMenuView menuView = (BottomNavigationMenuView) view.getChildAt(0);
    try {
        Field shiftingMode = menuView.getClass().getDeclaredField("mShiftingMode");
        shiftingMode.setAccessible(true);
        shiftingMode.setBoolean(menuView, false);
        shiftingMode.setAccessible(false);
        for (int i = 0; i < menuView.getChildCount(); i++) {
            BottomNavigationItemView item = (BottomNavigationItemView) menuView.getChildAt(i);
            //noinspection RestrictedApi
            item.setShiftingMode(false);
            // set once again checked value, so view will be updated
            //noinspection RestrictedApi
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("BNVHelper", "Unable to get shift mode field", e);
    } catch (IllegalAccessException e) {
        Log.e("BNVHelper", "Unable to change value of shift mode", e);
    }
}
```

This disables the Shifting behaviour of the menu when item count exceeds 3 nos.

USAGE

```
BottomNavigationView navView = (BottomNavigationView) findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Proguard Issue : Add following line proguard configuration file as well else, this wouldn't work.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
    boolean mShiftingMode;
}
```

或者，您可以创建一个类并从那里访问此方法。查看原始回复

注意：这是一个基于反射的HOTFIX，请在谷歌的支持库更新为直接调用函数后及时更新此处。

Alternatively, you can create a Class and access this method from there. [See Original Reply Here](#)

NOTE : This is a Reflection based **HOTFIX**, please update this once Google's support library is updated with a direct function call.

第27章：使用

SurfaceView进行画布绘制

第27.1节：带绘图线程的SurfaceView

本示例介绍如何创建一个带有专用绘图线程的SurfaceView。该实现还处理了制造商特定的问题等边界情况，以及启动/停止线程以节省CPU时间。

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
/**
 * 定义一个自定义SurfaceView类，用于处理绘图线程
 */
public class BaseSurface extends SurfaceView implements SurfaceHolder.Callback,
View.OnTouchListener, Runnable
{
    /**
     * 持有表面帧
     */
    private SurfaceHolder holder;

    /**
     * 绘图线程
     */
    private Thread drawThread;

    /**
     * 当画布准备好绘制时为真
     */
    private boolean surfaceReady = false;

    /**
     * 绘图线程标志
     */
    private boolean drawingActive = false;

    /**
     * 用于绘制示例矩形的画笔
     */
    private Paint samplePaint = new Paint();

    /**
     * 60帧每秒的每帧时间
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    public BaseSurface(Context context, AttributeSet attrs)
```

Chapter 27: Canvas drawing using SurfaceView

Section 27.1: SurfaceView with drawing thread

This example describes how to create a SurfaceView with a dedicated drawing thread. This implementation also handles edge cases such as manufacturer specific issues as well as starting/stopping the thread to save cpu time.

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
/**
 * Defines a custom SurfaceView class which handles the drawing thread
 */
public class BaseSurface extends SurfaceView implements SurfaceHolder.Callback,
View.OnTouchListener, Runnable
{
    /**
     * Holds the surface frame
     */
    private SurfaceHolder holder;

    /**
     * Draw thread
     */
    private Thread drawThread;

    /**
     * True when the surface is ready to draw
     */
    private boolean surfaceReady = false;

    /**
     * Drawing thread flag
     */
    private boolean drawingActive = false;

    /**
     * Paint for drawing the sample rectangle
     */
    private Paint samplePaint = new Paint();

    /**
     * Time per frame for 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    public BaseSurface(Context context, AttributeSet attrs)
```

```

{
    super(context, attrs);
    SurfaceHolder holder = getHolder();
    holder.addCallback(this);
    setOnTouchListener(this);

    // 红色
    samplePaint.setColor(0xffff0000);
    // 平滑边缘
    samplePaint.setAntiAlias(true);
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
    if (width == 0 || height == 0)
    {
        return;
    }

    // 调整你的用户界面大小
}

@Override
public void surfaceCreated(SurfaceHolder holder)
{
    this.holder = holder;

    if (drawThread != null)
    {
        Log.d(LOGTAG, "绘图线程仍在运行中..");
        drawingActive = false;
        try
        {
            drawThread.join();
        } catch (InterruptedException e)
        { // 什么也不做
        }
    }

    surfaceReady = true;
    startDrawThread();
    Log.d(LOGTAG, "已创建");
}

@Override
public void surfaceDestroyed(SurfaceHolder holder)
{
    // 不再使用Surface - 停止绘图线程
    stopDrawThread();
    // 并释放表面
    holder.getSurface().release();

    this.holder = null;
    surfaceReady = false;
    Log.d(LOGTAG, "Destroyed");
}

@Override
public boolean onTouch(View v, MotionEvent event)
{
    // 处理触摸事件
}

```

```

{
    super(context, attrs);
    SurfaceHolder holder = getHolder();
    holder.addCallback(this);
    setOnTouchListener(this);

    // red
    samplePaint.setColor(0xffff0000);
    // smooth edges
    samplePaint.setAntiAlias(true);
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
    if (width == 0 || height == 0)
    {
        return;
    }

    // resize your UI
}

@Override
public void surfaceCreated(SurfaceHolder holder)
{
    this.holder = holder;

    if (drawThread != null)
    {
        Log.d(LOGTAG, "draw thread still active..");
        drawingActive = false;
        try
        {
            drawThread.join();
        } catch (InterruptedException e)
        { // do nothing
        }
    }

    surfaceReady = true;
    startDrawThread();
    Log.d(LOGTAG, "Created");
}

@Override
public void surfaceDestroyed(SurfaceHolder holder)
{
    // Surface is not used anymore - stop the drawing thread
    stopDrawThread();
    // and release the surface
    holder.getSurface().release();

    this.holder = null;
    surfaceReady = false;
    Log.d(LOGTAG, "Destroyed");
}

@Override
public boolean onTouch(View v, MotionEvent event)
{
    // Handle touch events
}

```

```

    return true;
}

/**
* 停止绘图线程
*/
public void stopDrawThread()
{
    if (drawThread == null)
    {
        Log.d(LOGTAG, "DrawThread is null");
        return;
    }
    drawingActive = false;
    while (true)
    {
        try
        {
Log.d(LOGTAG, "请求最后一帧");
            drawThread.join(5000);
            break;
        } catch (Exception e)
        {
Log.e(LOGTAG, "无法与绘制线程合并");
        }
    }
    drawThread = null;
}

/**
* 创建一个新的绘制线程并启动它。
*/
public void startDrawThread()
{
    if (surfaceReady && drawThread == null)
    {
drawThread = new Thread(this, "绘制线程");
        drawingActive = true;
        drawThread.start();
    }
}

@Override
public void run()
{
Log.d(LOGTAG, "绘图线程已启动");
    long frameStartTime;
    long frameTime;

    /*
    * 为了在Nexus 7上可靠运行，我们在绘图线程开始时加入约500毫秒的延迟
    * (AOSP - 问题 58385)
    */
    if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
        android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
        android.os.Build.MODEL.equalsIgnoreCase("Nexus 7"))
    {
        Log.w(LOGTAG, "Sleep 500ms (设备：Asus Nexus 7)");
        try
        {
            Thread.sleep(500);
        } catch (InterruptedException ignored)
    }
}

```

```

    return true;
}

/**
* Stops the drawing thread
*/
public void stopDrawThread()
{
    if (drawThread == null)
    {
        Log.d(LOGTAG, "DrawThread is null");
        return;
    }
    drawingActive = false;
    while (true)
    {
        try
        {
Log.d(LOGTAG, "Request last frame");
            drawThread.join(5000);
            break;
        } catch (Exception e)
        {
            Log.e(LOGTAG, "Could not join with draw thread");
        }
    }
    drawThread = null;
}

/**
* Creates a new draw thread and starts it.
*/
public void startDrawThread()
{
    if (surfaceReady && drawThread == null)
    {
        drawThread = new Thread(this, "Draw thread");
        drawingActive = true;
        drawThread.start();
    }
}

@Override
public void run()
{
    Log.d(LOGTAG, "Draw thread started");
    long frameStartTime;
    long frameTime;

    /*
    * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing thread
    * (AOSP - Issue 58385)
    */
    if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
        android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
        android.os.Build.MODEL.equalsIgnoreCase("Nexus 7"))
    {
        Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
        try
        {
            Thread.sleep(500);
        } catch (InterruptedException ignored)
    }
}

```

```

        }

    }
    try
    {
        while (drawingActive)
        {
            if (holder == null)
            {
                return;
            }

frameStartTime = System.nanoTime();
Canvas canvas = holder.lockCanvas();
if (canvas != null)
{
    // 使用黑色清屏
canvas.drawRGB(255, 0, 0, 0);

        try
        {
            // 在此处绘制内容
canvas.drawRect(0, 0, getWidth() / 2, getHeight() / 2, samplePaint);
        } finally
        {

holder.unlockCanvasAndPost(canvas);
        }
    }

    // 计算绘制帧所需的时间 (毫秒)
frameTime = (System.nanoTime() - frameStartTime) / 0000000;

    if (frameTime < MAX_FRAME_TIME) // 比最大帧率快 - 限制FPS
    {
        try
        {
            Thread.sleep(MAX_FRAME_TIME - frameTime);
        } catch (InterruptedException e)
        {
            // 忽略
        }
    }
} catch (Exception e)
{
Log.w(LOGTAG, "锁定/解锁时异常");
}
Log.d(LOGTAG, "绘制线程结束");
}
}

```

此布局仅包含自定义的SurfaceView，并将其最大化到屏幕大小。

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sample.devcore.org.surfaceviewsample.MainActivity">

```

```

        }

    }
    try
    {
        while (drawingActive)
        {
            if (holder == null)
            {
                return;
            }

frameStartTime = System.nanoTime();
Canvas canvas = holder.lockCanvas();
if (canvas != null)
{
    // clear the screen using black
canvas.drawRGB(255, 0, 0, 0);

        try
        {
            // Your drawing here
            canvas.drawRect(0, 0, getWidth() / 2, getHeight() / 2, samplePaint);
        } finally
        {

holder.unlockCanvasAndPost(canvas);
        }
    }

    // calculate the time required to draw the frame in ms
frameTime = (System.nanoTime() - frameStartTime) / 1000000;

    if (frameTime < MAX_FRAME_TIME) // faster than the max fps - limit the FPS
    {
        try
        {
            Thread.sleep(MAX_FRAME_TIME - frameTime);
        } catch (InterruptedException e)
        {
            // ignore
        }
    }
} catch (Exception e)
{
    Log.w(LOGTAG, "Exception while locking/unlocking");
}
Log.d(LOGTAG, "Draw thread finished");
}
}

```

This layout only contains the custom SurfaceView and maximizes it to the screen size.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sample.devcore.org.surfaceviewsample.MainActivity">

```

```
<sample.devcore.org.surfaceviewsample.BaseSurface
    android:id="@+id/baseSurface"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
</LinearLayout>
```

使用SurfaceView的活动负责启动和停止绘图线程。这种方法节省电池，因为一旦活动进入后台，绘图就会停止。

```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{

    /**
     * Surface对象
     */
    private BaseSurface surface;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        surface = (BaseSurface) findViewById(R.id.baseSurface);
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        // 启动绘图
        surface.startDrawThread();
    }

    @Override
    protected void onPause()
    {
        // 停止绘图以节省CPU时间
        surface.stopDrawThread();
        super.onPause();
    }
}
```

```
<sample.devcore.org.surfaceviewsample.BaseSurface
    android:id="@+id/baseSurface"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
</LinearLayout>
```

The activity which uses the SurfaceView is responsible for starting and stopping the drawing thread. This approach saves battery as the drawing is stopped as soon as the activity gets in the background.

```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{

    /**
     * Surface object
     */
    private BaseSurface surface;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        surface = (BaseSurface) findViewById(R.id.baseSurface);
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        // start the drawing
        surface.startDrawThread();
    }

    @Override
    protected void onPause()
    {
        // stop the drawing to save cpu time
        surface.stopDrawThread();
        super.onPause();
    }
}
```

第28章：创建自定义视图

第28.1节：创建自定义视图

如果你需要一个完全自定义的视图，你需要继承 `View` (所有Android视图的超类) 并提供你自定义的测量 (`onMeasure(...)`) 和绘制 (`onDraw(...)`) 方法：

1. 创建你的自定义视图骨架：这基本上适用于每个自定义视图。这里我们创建一个可以绘制笑脸的自定义视图骨架，称为SmileyView：

```
public class SmileyView extends View {  
    private Paint mCirclePaint;  
    private Paint mEyeAndMouthPaint;  
  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius;  
    private RectF mArcBounds = new RectF();  
  
    public SmileyView(Context context) {  
        this(context, null, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        this(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
        initPaints();  
    }  
  
    private void initPaints() /* ... */  
  
    @Override  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) /* ... */  
  
    @Override  
    protected void onDraw(Canvas canvas) /* ... */  
}
```

2. 初始化你的画笔：`Paint`对象是你虚拟画布上的画刷，定义了几何图形的渲染方式（例如颜色、填充和描边样式等）。这里我们创建了两个`Paint`对象，一个是用于圆形的黄色填充画笔，另一个是用于眼睛和嘴巴的黑色描边画笔：

```
private void initPaints() {  
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
    mCirclePaint.setStyle(Paint.Style.FILL);  
    mCirclePaint.setColor(Color.YELLOW);  
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);  
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);  
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);  
    mEyeAndMouthPaint.setColor(Color.BLACK);  
}
```

3. 实现你自己的 `onMeasure(...)` 方法：这是必须的，以便父布局（例如

Chapter 28: Creating Custom Views

Section 28.1: Creating Custom Views

If you need a completely customized view, you'll need to subclass `View` (the superclass of all Android views) and provide your custom sizing (`onMeasure(...)`) and drawing (`onDraw(...)`) methods:

1. **Create your custom view skeleton:** this is basically the same for every custom view. Here we create the skeleton for a custom view that can draw a smiley, called `SmileyView`:

```
public class SmileyView extends View {  
    private Paint mCirclePaint;  
    private Paint mEyeAndMouthPaint;  
  
    private float mCenterX;  
    private float mCenterY;  
    private float mRadius;  
    private RectF mArcBounds = new RectF();  
  
    public SmileyView(Context context) {  
        this(context, null, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs) {  
        this(context, attrs, 0);  
    }  
  
    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
        initPaints();  
    }  
  
    private void initPaints() /* ... */  
  
    @Override  
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) /* ... */  
  
    @Override  
    protected void onDraw(Canvas canvas) /* ... */  
}
```

2. **Initialize your paints:** the `Paint` objects are the brushes of your virtual canvas defining how your geometric objects are rendered (e.g. color, fill and stroke style, etc.). Here we create two `Paints`, one yellow filled paint for the circle and one black stroke paint for the eyes and the mouth:

```
private void initPaints() {  
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
    mCirclePaint.setStyle(Paint.Style.FILL);  
    mCirclePaint.setColor(Color.YELLOW);  
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);  
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);  
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);  
    mEyeAndMouthPaint.setColor(Color.BLACK);  
}
```

3. **Implement your own `onMeasure(...)` method:** this is required so that the parent layouts (e.g.

FrameLayout) 能够正确对齐你的自定义视图。它提供了一组 measureSpecs，你可以用来确定视图的高度和宽度。这里我们通过确保高度和宽度相同来创建一个正方形：

```
@Override  
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {  
    int w = MeasureSpec.getSize(widthMeasureSpec);  
    int h = MeasureSpec.getSize(heightMeasureSpec);  
  
    int size = Math.min(w, h);  
    setMeasuredDimension(size, size);  
}
```

注意 onMeasure(...) 方法中必须至少调用一次 setMeasuredDimension(..)，否则你的自定义视图会因 IllegalStateException 而崩溃。

4. 实现你自己的 onSizeChanged(...) 方法：这允许你获取当前自定义视图的高度和宽度，以便正确调整渲染代码。这里我们只是计算中心点和半径：你的自定义视图，以便正确调整渲染代码。这里我们只是计算中心点和半径：

```
@Override  
protected void onSizeChanged(int w, int h, int oldw, int oldh) {  
    mCenterX = w / 2f;  
    mCenterY = h / 2f;  
    mRadius = Math.min(w, h) / 2f;  
}
```

5. 实现你自己的 onDraw(...) 方法：这是你实现视图实际渲染的地方。
它提供了一个 Canvas 对象，你可以在其上绘制（有关所有可用绘制方法，请参阅官方 Canvas 文档）。

```
@Override  
protected void onDraw(Canvas canvas) {  
    // 绘制脸部  
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);  
    // 绘制眼睛  
    float eyeRadius = mRadius / 5f;  
    float eyeOffsetX = mRadius / 3f;  
    float eyeOffsetY = mRadius / 3f;  
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,  
        mEyeAndMouthPaint);  
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,  
        mEyeAndMouthPaint);  
    // 绘制嘴巴  
    float mouthInset = mRadius / 3f;  
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -  
        mouthInset);  
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);  
}
```

6. 将您的自定义视图添加到布局中：现在可以将自定义视图包含在您拥有的任何布局文件中。
这里我们只是将它包裹在一个FrameLayout中：

```
<FrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">
```

FrameLayout) can properly align your custom view. It provides a set of measureSpecs that you can use to determine your view's height and width. Here we create a square by making sure that the height and width are the same:

```
@Override  
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {  
    int w = MeasureSpec.getSize(widthMeasureSpec);  
    int h = MeasureSpec.getSize(heightMeasureSpec);  
  
    int size = Math.min(w, h);  
    setMeasuredDimension(size, size);  
}
```

Note that onMeasure(...) must contain at least one call to setMeasuredDimension(..) or else your custom view will crash with an [IllegalStateException](#).

4. **Implement your own onSizeChanged(...)** method: this allows you to catch the current height and width of your custom view to properly adjust your rendering code. Here we just calculate our center and our radius:

```
@Override  
protected void onSizeChanged(int w, int h, int oldw, int oldh) {  
    mCenterX = w / 2f;  
    mCenterY = h / 2f;  
    mRadius = Math.min(w, h) / 2f;  
}
```

5. **Implement your own onDraw(...)** method: this is where you implement the actual rendering of your view.
It provides a [Canvas](#) object that you can draw on (see the official [Canvas](#) documentation for all drawing methods available).

```
@Override  
protected void onDraw(Canvas canvas) {  
    // draw face  
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);  
    // draw eyes  
    float eyeRadius = mRadius / 5f;  
    float eyeOffsetX = mRadius / 3f;  
    float eyeOffsetY = mRadius / 3f;  
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,  
        mEyeAndMouthPaint);  
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,  
        mEyeAndMouthPaint);  
    // draw mouth  
    float mouthInset = mRadius / 3f;  
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -  
        mouthInset);  
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);  
}
```

6. **Add your custom view to a layout:** the custom view can now be included in any layout files that you have.
Here we just wrap it inside a FrameLayout:

```
<FrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">
```

```

<com.example.app.SmileyView
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</FrameLayout>

```

请注意，建议在视图代码完成后构建您的项目。未构建项目时，您将无法在Android Studio的预览屏幕上看到该视图。

将所有内容整合后，启动包含上述布局的活动时，您应该会看到以下屏幕：



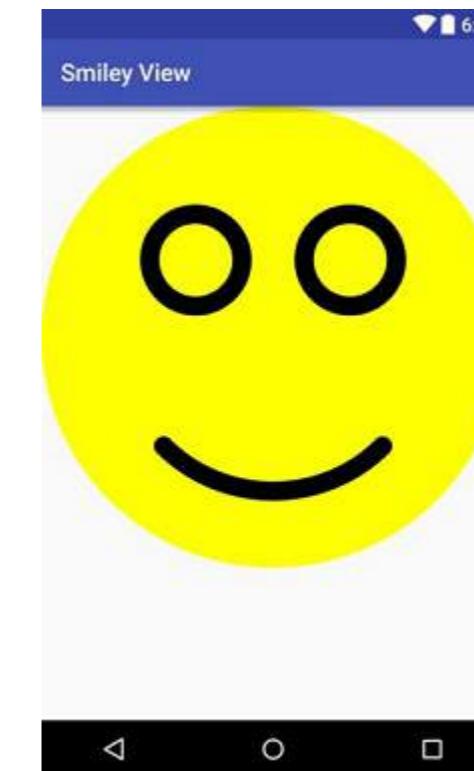
```

<com.example.app.SmileyView
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</FrameLayout>

```

Note that it is recommended to build your project after the view code is finished. Without building it you won't be able to see the view on a preview screen in Android Studio.

After putting everything together, you should be greeted with the following screen after launching the activity containing the above layout:



第28.2节：向视图添加属性

自定义视图也可以使用自定义属性，这些属性可以在Android布局资源文件中使用。要向自定义视图添加属性，您需要执行以下操作：

1. 定义属性的名称和类型：这在res/values/attrs.xml文件中完成（如有必要，请创建该文件）。

下面的文件为我们的笑脸的面部颜色定义了一个颜色属性，为笑脸的表情定义了一个枚举属性：

```

<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- 其他视图的属性 -->
</resources>

```

2. 在布局中使用您的属性：这可以在使用自定义视图的任何布局文件中完成。

下面的布局文件创建了一个带有快乐黄色笑脸的屏幕：

Section 28.2: Adding attributes to views

Custom views can also take custom attributes which can be used in Android layout resource files. To add attributes to your custom view you need to do the following:

- Define the name and type of your attributes:** this is done inside res/values/attrs.xml (create it if necessary). The following file defines a color attribute for our smiley's face color and an enum attribute for the smiley's expression:

```

<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- attributes for other views -->
</resources>

```

- Use your attributes inside your layout:** this can be done inside any layout files that use your custom view.

The following layout file creates a screen with a happy yellow smiley:

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>

```

提示：自定义属性在 Android Studio 2.1 及更早版本（以及可能的未来版本）中，使用tools:前缀时不起作用。在此示例中，将app:smileyColor替换为tools:smileyColor会导致smileyColor既不会在运行时设置，也不会在设计时设置。

3. 读取你的属性：这在你的自定义视图源代码中完成。以下代码片段 SmileyView 演示了如何提取属性：

```

public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
            defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
            Expression.HAPPY);
        // 重要：始终回收 TypedArray
        a.recycle();
        // 初始化画笔(); ...
    }
}

```

4. (可选) 添加默认样式：这是通过添加一个带有默认值的样式并在自定义视图中加载它来完成的自定义视图。以下默认的笑容样式表示一个快乐的黄色笑脸：

```

<!-- styles.xml --&gt;
&lt;style name="DefaultSmileyStyle"&gt;
    &lt;item name="smileyColor"&gt;#ffff00&lt;/item&gt;
    &lt;item name="smileyExpression"&gt;happy&lt;/item&gt;
&lt;/style&gt;
</pre>

```

该样式通过在调用

obtainStyledAttributes 时作为最后一个参数添加到我们的 SmileyView 中来应用（见步骤 3 中的代码）：

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>

```

Tip: Custom attributes do not work with the tools: prefix in Android Studio 2.1 and older (and possibly in future versions). In this example, replacing app:smileyColor with tools:smileyColor would result in smileyColor neither being set during runtime nor at design time.

3. **Read your attributes:** this is done inside your custom view source code. The following snippet of SmileyView demonstrates how the attributes can be extracted:

```

public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
            defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
            Expression.HAPPY);
        // Important: always recycle the TypedArray
        a.recycle();

        // initPaints(); ...
    }
}

```

4. **(Optional) Add default style:** this is done by adding a style with the default values and loading it inside your custom view. The following default smiley style represents a happy yellow one:

```

<!-- styles.xml --&gt;
&lt;style name="DefaultSmileyStyle"&gt;
    &lt;item name="smileyColor"&gt;#ffff00&lt;/item&gt;
    &lt;item name="smileyExpression"&gt;happy&lt;/item&gt;
&lt;/style&gt;
</pre>

```

Which gets applied in our SmileyView by adding it as the last parameter of the call to obtainStyledAttributes (see code in step 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView, defStyleAttr, R.styleable.DefaultSmileyViewStyle);
```

请注意，在膨胀的布局文件中设置的任何属性值（参见步骤2中的代码）将覆盖默认样式的对应值。

5. (可选) 在主题中提供样式：这是通过添加一个新的样式引用属性来完成的，该属性可以在主题中使用，并为该属性提供样式。这里我们简单地将引用属性命名为smileyStyle：

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

然后我们在应用主题中为其提供样式（这里我们只是重用步骤4中的默认样式）：

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

第28.3节：自定义视图性能提示

不要在onDraw中分配新对象

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //不要在这里分配内存
}
```

不要在画布上绘制drawable...

```
drawable.setBounds(boundsRect);
drawable.draw(canvas);
```

使用Bitmap以加快绘制速度：

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

不要重绘整个视图来更新其中一小部分。应当只重绘视图的特定部分。

```
invalidate(boundToBeRefreshed);
```

如果你的视图正在进行连续动画，例如显示每秒的表盘，至少应在活动的onStop()方法中停止动画，并在活动的onStart()方法中重新启动动画。

不要在视图的onDraw方法中进行任何计算，应该在调用invalidate()之前完成绘制。通过使用此技术，可以避免视图的帧丢失。

旋转

视图的基本操作包括平移、旋转等.....几乎每个开发者在使用自定义视图时都遇到过这个问题

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView, defStyleAttr, R.styleable.DefaultSmileyViewStyle);
```

Note that any attribute values set in the inflated layout file (see code in step 2) will override the corresponding values of the default style.

5. (Optional) Provide styles inside themes: this is done by adding a new style reference attribute which can be used inside your themes and providing a style for that attribute. Here we simply name our reference attribute smileyStyle:

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

Which we then provide a style for in our app theme (here we just reuse the default style from step 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Section 28.3: CustomView performance tips

Do not allocate new objects in **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Instead of drawing drawables in canvas...

```
drawable.setBounds(boundsRect);
drawable.draw(canvas);
```

Use a Bitmap for faster drawing:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Do not redraw the entire view to update just a small part of it. Instead redraw the specific part of view.

```
invalidate(boundToBeRefreshed);
```

If your view is doing some continuous animation, for instance a watch-face showing each and every second, at least stop the animation at onStop() of the activity and start it back on onStart() of the activity.

Do not do any calculations inside the onDraw method of a view, you should instead finish drawing before calling invalidate(). By using this technique you can avoid frame dropping in your view.

Rotations

The basic operations of a view are translate, rotate, etc... Almost every developer has faced this problem when they

当视图需要显示旋转后的内容且位图必须在该自定义视图中旋转时，许多人会认为这很耗费资源。很多人认为旋转位图非常昂贵，因为为此需要转换位图的像素矩阵。但事实并非如此！与其旋转位图，不如直接旋转画布本身！

```
// 保存画布状态
int save = canvas.save();
// 通过提供中心点作为旋转轴和角度来旋转画布
canvas.rotate(pivotX, pivotY, angle);
// 绘制你想要的内容
// 基本上你在这里绘制的任何内容都会按照你旋转画布的角度来绘制
canvas.drawBitmap(...);
// 现在将画布恢复到原始状态
canvas.restore(save);
// 除非画布恢复到其原始状态，否则后续的绘制也会被旋转。
```

第28.4节：创建复合视图

复合视图（compound view）是一个自定义的视图组（ViewGroup），在外围程序代码中被视为单个视图。这样的 ViewGroup 在类似领域驱动设计（DDD）的设计中非常有用，因为它可以对应一个聚合体，在本例中是一个联系人（Contact）。它可以在显示联系人信息的任何地方重复使用。

这意味着外围的控制器代码，如Activity、Fragment或Adapter，可以简单地将数据对象传递给视图，而无需将其拆分成多个不同的UI控件。

这有助于代码重用，并根据SOLID原则实现更好的设计。

布局XML

这通常是你开始的地方。你有一段现有的XML，发现自己经常重复使用，可能作为一个 `<include>`。将其提取到一个单独的XML文件中，并用`<merge>`元素包裹根标签：

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/photo"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_alignParentRight="true" />

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/photo" />

    <TextView
        android:id="@+id/phone_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/name"
        android:layout_toLeftOf="@+id/photo" />
</merge>
```

此XML文件在Android Studio的布局编辑器中运行良好。你可以像对待其他布局一样对待它。

use bitmap or gradients in their custom view. If the view is going to show a rotated view and the bitmap has to be rotated in that custom view, many of us will think that it will be expensive. Many think that rotating a bitmap is very expensive because in order to do that, you need to translate the bitmap's pixel matrix. But the truth is that it is not that tough! Instead of rotating the bitmap, just rotate the canvas itself!

```
// Save the canvas state
int save = canvas.save();
// Rotate the canvas by providing the center point as pivot and angle
canvas.rotate(pivotX, pivotY, angle);
// Draw whatever you want
// Basically whatever you draw here will be drawn as per the angle you rotated the canvas
canvas.drawBitmap(...);
// Now restore your your canvas to its original state
canvas.restore(save);
// Unless canvas is restored to its original state, further draw will also be rotated.
```

Section 28.4: Creating a compound view

A **compound view** is a custom ViewGroup that's treated as a single view by the surrounding program code. Such a ViewGroup can be really useful in DDD-like design, because it can correspond to an aggregate, in this example, a Contact. It can be reused everywhere that contact is displayed.

This means that the surrounding controller code, an Activity, Fragment or Adapter, can simply pass the data object to the view without picking it apart into a number of different UI widgets.

This facilitates code reuse and makes for a better design according to [SOLID principles](#).

The layout XML

This is usually where you start. You have an existing bit of XML that you find yourself reusing, perhaps as an `<include>`. Extract it into a separate XML file and wrap the root tag in a `<merge>` element:

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/photo"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_alignParentRight="true" />

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/photo" />

    <TextView
        android:id="@+id/phone_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/name"
        android:layout_toLeftOf="@+id/photo" />
</merge>
```

This XML file keeps working in the Layout Editor in Android Studio perfectly fine. You can treat it like any other

布局。

复合视图组

获得XML文件后，创建自定义视图组。

```
import android.annotation.TargetApi;
import android.content.Context;
import android.os.Build;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.ImageView;
import android.widget.TextView;

import myapp.R;

/**
 * 一个用于显示联系人信息的复合视图。
 *
 * 该类可以放入XML布局中或通过编程方式实例化，
 * 两种方式均能正常工作。
 */
public class ContactView extends RelativeLayout {

    // 该类继承自RelativeLayout，因为它自带对子项的自动// (MATCH_PARENT, MATCH_PARENT) 布局。如果需要更多控制，// 可以继承原始的android.view.ViewGroup类。请参阅布局XML 中的注释，// 说明为什么不建议继承像RelativeLayout这样复杂的视图。

    // 1. 实现父类构造函数。
    public ContactView(Context context) {
        super(context);
        init(context, null);
    }

    // 为简化示例，省略了另外两个构造函数

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public ContactView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(context, attrs);
    }

    // 2. 通过使用'this'作为父视图，初始化视图并加载XML布局
    private TextView mName;
    private TextView mPhoneNumber;
    private ImageView mPhoto;

    private void init(Context context, AttributeSet attrs) {
        LayoutInflater.from(context).inflate(R.layout.contact_view, this, true);
        mName = (TextView) findViewById(R.id.name);
        mPhoneNumber = (TextView) findViewById(R.id.phone_number);
        mPhoto = (ImageView) findViewById(R.id.photo);
    }

    // 3. 定义一个在你的领域模型中表达的设置器。这就是示例的全部内容。
    // 所有控制器代码都可以调用这个设置器，而不是去处理
    // 大量的字符串、可见性选项、颜色、动画等。如果你不使用自定义视图，
```

layout.

The compound ViewGroup

Once you have the XML file, create the custom view group.

```
import android.annotation.TargetApi;
import android.content.Context;
import android.os.Build;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.ImageView;
import android.widget.TextView;
import myapp.R;

/**
 * A compound view to show contacts.
 *
 * This class can be put into an XML layout or instantiated programmatically, it
 * will work correctly either way.
 */
public class ContactView extends RelativeLayout {

    // This class extends RelativeLayout because that comes with an automatic
    // (MATCH_PARENT, MATCH_PARENT) layout for its child item. You can extend
    // the raw android.view.ViewGroup class if you want more control. See the
    // note in the layout XML why you wouldn't want to extend a complex view
    // such as RelativeLayout.

    // 1. Implement superclass constructors.
    public ContactView(Context context) {
        super(context);
        init(context, null);
    }

    // two extra constructors left out to keep the example shorter

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public ContactView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(context, attrs);
    }

    // 2. Initialize the view by inflating an XML using 'this' as parent
    private TextView mName;
    private TextView mPhoneNumber;
    private ImageView mPhoto;

    private void init(Context context, AttributeSet attrs) {
        LayoutInflater.from(context).inflate(R.layout.contact_view, this, true);
        mName = (TextView) findViewById(R.id.name);
        mPhoneNumber = (TextView) findViewById(R.id.phone_number);
        mPhoto = (ImageView) findViewById(R.id.photo);
    }

    // 3. Define a setter that's expressed in your domain model. This is what the example is
    // all about. All controller code can just invoke this setter instead of fiddling with
    // lots of strings, visibility options, colors, animations, etc. If you don't use a
```

```
// 这段代码通常会出现在静态辅助方法中（不好的做法），或者这段代码会被复制粘贴到各处（更糟）。
public void setContact(Contact contact) {
    mName.setText(contact.getName());
    mPhoneNumber.setText(contact.getPhoneNumber());
    if (contact.hasPhoto()) {
        mPhoto.setVisibility(View.VISIBLE);
        mPhoto.setImageBitmap(contact.getPhoto());
    } else {
        mPhoto.setVisibility(View.GONE);
    }
}
```

init(Context, AttributeSet)方法是您读取任何自定义XML属性的地方，如“向视图添加属性”中所述。

有了这些部分，您就可以在应用中使用它了。

XML中的用法

这是一个示例fragment_contact_info.xml，展示了如何将单个ContactView放置在消息列表顶部：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- 复合视图就像其他视图XML元素一样 -->
    <myapp.ContactView
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/message_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

</LinearLayout>
```

代码中的用法

这是一个示例RecyclerView.Adapter，显示联系人列表。这个示例说明当控制器代码完全不涉及视图操作时，代码会变得多么简洁。

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;
    }
```

```
// custom view, this code will usually end up in a static helper method (bad) or copies
// of this code will be copy-pasted all over the place (worse).
public void setContact(Contact contact) {
    mName.setText(contact.getName());
    mPhoneNumber.setText(contact.getPhoneNumber());
    if (contact.hasPhoto()) {
        mPhoto.setVisibility(View.VISIBLE);
        mPhoto.setImageBitmap(contact.getPhoto());
    } else {
        mPhoto.setVisibility(View.GONE);
    }
}
```

The `init(Context, AttributeSet)` method is where you would read any custom XML attributes as explained in Adding Attributes to Views.

With these pieces in place, you can use it in your app.

Usage in XML

Here's an example `fragment_contact_info.xml` that illustrates how you'd put a single ContactView on top of a list of messages:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- The compound view becomes like any other view XML element -->
    <myapp.ContactView
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/message_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

</LinearLayout>
```

Usage in Code

Here's an example `RecyclerView.Adapter` that shows a list of contacts. This example illustrates just how much cleaner the controller code gets when it's completely free of View manipulation.

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;
    }
```

```

}

@Override
public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    ContactView v = new ContactView(context); // <--- 这里
    return new ContactsViewHolder(v);
}

@Override
public void onBindViewHolder(ContactsViewHolder holder, int position) {
    Contact contact = this.getItem(position);
    holder.setContact(contact); // <--- 这里
}

static class ContactsViewHolder extends RecyclerView.ViewHolder {

    public ContactsViewHolder(ContactView itemView) {
        super(itemView);
    }

    public void setContact(Contact contact) {
        ((ContactView) itemView).setContact(contact); // <--- 这里
    }
}

```

```

}

@Override
public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    ContactView v = new ContactView(context); // <--- this
    return new ContactsViewHolder(v);
}

@Override
public void onBindViewHolder(ContactsViewHolder holder, int position) {
    Contact contact = this.getItem(position);
    holder.setContact(contact); // <--- this
}

static class ContactsViewHolder extends RecyclerView.ViewHolder {

    public ContactsViewHolder(ContactView itemView) {
        super(itemView);
    }

    public void setContact(Contact contact) {
        ((ContactView) itemView).setContact(contact); // <--- this
    }
}

```

第28.5节：作为drawableRight的SVG/VectorDrawable复合视图

开发此复合视图的主要动机是，5.0以下的设备不支持TextView/EditText中drawable内的SVG。另一个优点是，我们可以设置EditText中drawableRight的高度和宽度。我已将其从我的项目中分离出来，并创建了一个独立模块。模块名称：custom_edit_drawable（前缀简称：c_d_e）

使用“c_d_e_”前缀，以防止应用模块资源被误覆盖。例如：Google在支持库中使用“abc”前缀。

build.gradle

```

dependencies {
    compile 'com.android.support:appcompat-v7:25.3.1'
}

```

使用AppCompat版本>= 23

布局文件：c_e_d_compound_view.xml

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/edt_search"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:maxLines="1"/>

```

Section 28.5: Compound view for SVG/VectorDrawable as drawableRight

Main motive to develop this compound view is, below 5.0 devices does not support svg in drawable inside TextView/EditText. One more pros is, we can set height and width of drawableRight inside EditText. I have separated it from my project and created in separate module. **Module Name : custom_edit_drawable (short name for prefix- c_d_e)**

"c_d_e_" prefix to use so that app module resources should not override them by mistake. Example : "abc" prefix is used by google in support library.

build.gradle

```

dependencies {
    compile 'com.android.support:appcompat-v7:25.3.1'
}

```

use AppCompat >= 23

Layout file : c_e_d_compound_view.xml

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/edt_search"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:maxLines="1"/>

```

```

    android:paddingEnd="40dp"
    android:paddingLeft="5dp"
    android:paddingRight="40dp"
    android:paddingStart="5dp" />

<!--确保你没有使用ImageView来代替这个-->
<android.support.v7.widget.AppCompatImageView
    android:id="@+id/drawableRight_search"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_gravity="right|center_vertical"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp" />
</FrameLayout>

```

自定义属性 : attrs.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="EditTextWithDrawable">
        <attr name="c_e_d_drawableRightSVG" format="reference" />
        <attr name="c_e_d_hint" format="string" />
        <attr name="c_e_d_textSize" format="dimension" />
        <attr name="c_e_d_textColor" format="color" />
    </declare-styleable>
</resources>

```

代码 : EditTextWithDrawable.java

```

public class EditTextWithDrawable extends FrameLayout {
    public AppCompatImageView mDrawableRight;
    public EditText mEditText;

    public EditTextWithDrawable(Context context) {
        super(context);
        init(null);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init(attrs);
    }

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(attrs);
    }

    private void init(AttributeSet attrs) {
        if (attrs != null && !isInEditMode()) {
            LayoutInflator inflater = (LayoutInflator) getContext()
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflater.inflate(R.layout.c_e_d_compound_view, this, true);
            mDrawableRight = (AppCompatImageView) ((FrameLayout) getChildAt(0)).getChildAt(1);
        }
    }
}

```

```

    android:paddingEnd="40dp"
    android:paddingLeft="5dp"
    android:paddingRight="40dp"
    android:paddingStart="5dp" />

```

```

<!--make sure you are not using ImageView instead of this-->
<android.support.v7.widget.AppCompatImageView
    android:id="@+id/drawableRight_search"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_gravity="right|center_vertical"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp" />
</FrameLayout>

```

Custom Attributes : attrs.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="EditTextWithDrawable">
        <attr name="c_e_d_drawableRightSVG" format="reference" />
        <attr name="c_e_d_hint" format="string" />
        <attr name="c_e_d_textSize" format="dimension" />
        <attr name="c_e_d_textColor" format="color" />
    </declare-styleable>
</resources>

```

Code : EditTextWithDrawable.java

```

public class EditTextWithDrawable extends FrameLayout {
    public AppCompatImageView mDrawableRight;
    public EditText mEditText;

    public EditTextWithDrawable(Context context) {
        super(context);
        init(null);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init(attrs);
    }

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(attrs);
    }

    private void init(AttributeSet attrs) {
        if (attrs != null && !isInEditMode()) {
            LayoutInflator inflater = (LayoutInflator) getContext()
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflater.inflate(R.layout.c_e_d_compound_view, this, true);
            mDrawableRight = (AppCompatImageView) ((FrameLayout) getChildAt(0)).getChildAt(1);
        }
    }
}

```

```

mEditText = (EditText) ((FrameLayout) getChildAt(0)).getChildAt(0);

TypedArray attributeArray = getContext().obtainStyledAttributes(
    attrs,
R.styleable.EditTextWithDrawable);

int drawableRes =
attributeArray.getResourceId(
R.styleable.EditTextWithDrawable_c_e_d_drawableRightSVG, -1);
if (drawableRes != -1) {
mDrawableRight.setImageResource(drawableRes);
}

mEditText.setHint(attributeArray.getString(
R.styleable.EditTextWithDrawable_c_e_d_hint));
mEditText.setTextColor(attributeArray.getColor(
R.styleable.EditTextWithDrawable_c_e_d_textColor, Color.BLACK));
int textSize =
attributeArray.getDimensionPixelSize(R.styleable.EditTextWithDrawable_c_e_d_textSize, 15);
mEditText.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
android.ViewGroup.LayoutParams layoutParams = mDrawableRight.getLayoutParams();
layoutParams.width = (textSize * 3) / 2;
layoutParams.height = (textSize * 3) / 2;
mDrawableRight.setLayoutParams(layoutParams);

attributeArray.recycle();
}
}
}

```

示例：如何使用上述视图

布局：activity_main.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <com.customeditdrawable.AppEditTextWithDrawable
        android:id="@+id edt_search_emp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:c_e_d_drawableRightSVG="@drawable/ic_svg_search"
        app:c_e_d_hint="@string/hint_search_here"
        app:c_e_d_textColor="@color/text_color_dark_on_light_bg"
        app:c_e_d_textSize="@dimen/text_size_small" />
</LinearLayout>

```

活动：MainActivity.java

```

public class MainActivity extends AppCompatActivity {
    EditTextWithDrawable mEditTextWithDrawable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mEditTextWithDrawable= (EditTextWithDrawable) findViewById(R.id.edt_search_emp);
    }
}

```

```

mEditText = (EditText) ((FrameLayout) getChildAt(0)).getChildAt(0);

TypedArray attributeArray = getContext().obtainStyledAttributes(
    attrs,
R.styleable.EditTextWithDrawable);

int drawableRes =
attributeArray.getResourceId(
R.styleable.EditTextWithDrawable_c_e_d_drawableRightSVG, -1);
if (drawableRes != -1) {
mDrawableRight.setImageResource(drawableRes);
}

mEditText.setHint(attributeArray.getString(
R.styleable.EditTextWithDrawable_c_e_d_hint));
mEditText.setTextColor(attributeArray.getColor(
R.styleable.EditTextWithDrawable_c_e_d_textColor, Color.BLACK));
int textSize =
attributeArray.getDimensionPixelSize(R.styleable.EditTextWithDrawable_c_e_d_textSize, 15);
mEditText.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
android.ViewGroup.LayoutParams layoutParams = mDrawableRight.getLayoutParams();
layoutParams.width = (textSize * 3) / 2;
layoutParams.height = (textSize * 3) / 2;
mDrawableRight.setLayoutParams(layoutParams);

attributeArray.recycle();
}
}
}

```

Example : How to use above view

Layout : activity_main.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <com.customeditdrawable.AppEditTextWithDrawable
        android:id="@+id edt_search_emp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:c_e_d_drawableRightSVG="@drawable/ic_svg_search"
        app:c_e_d_hint="@string/hint_search_here"
        app:c_e_d_textColor="@color/text_color_dark_on_light_bg"
        app:c_e_d_textSize="@dimen/text_size_small" />
</LinearLayout>

```

Activity : MainActivity.java

```

public class MainActivity extends AppCompatActivity {
    EditTextWithDrawable mEditTextWithDrawable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mEditTextWithDrawable= (EditTextWithDrawable) findViewById(R.id.edt_search_emp);
    }
}

```

第28.6节：响应触摸事件

许多自定义视图需要以触摸事件的形式接受用户交互。你可以通过重写 `onTouchEvent` 来获取触摸事件。有许多动作可以过滤。主要的有

- ACTION_DOWN：当你的手指首次触摸视图时触发一次。
- ACTION_MOVE：每当你的手指在视图上稍微移动时调用。它会被调用多次。
- ACTION_UP：当你抬起手指离开屏幕时调用的最后一个动作。

你可以将以下方法添加到你的视图中，然后在触摸并移动手指时观察日志输出。

```
@Override  
public boolean onTouchEvent(MotionEvent event) {  
  
    int x = (int) event.getX();  
    int y = (int) event.getY();  
    int action = event.getAction();  
  
    switch (action) {  
        case MotionEvent.ACTION_DOWN:  
Log.i("CustomView", "onTouchEvent: ACTION_DOWN: x = " + x + ", y = " + y);  
        break;  
  
        case MotionEvent.ACTION_MOVE:  
Log.i("CustomView", "onTouchEvent: ACTION_MOVE: x = " + x + ", y = " + y);  
        break;  
  
        case MotionEvent.ACTION_UP:  
Log.i("CustomView", "onTouchEvent: ACTION_UP: x = " + x + ", y = " + y);  
        break;  
    }  
    return true;  
}
```

进一步阅读：

- [Android 官方文档：响应触摸事件](#)

Section 28.6: Responding to Touch Events

Many custom views need to accept user interaction in the form of touch events. You can get access to touch events by overriding `onTouchEvent`. There are a number of actions you can filter out. The main ones are

- ACTION_DOWN: This is triggered once when your finger first touches the view.
- ACTION_MOVE: This is called every time your finger moves a little across the view. It gets called many times.
- ACTION_UP: This is the last action to be called as you lift your finger off the screen.

You can add the following method to your view and then observe the log output when you touch and move your finger around your view.

```
@Override  
public boolean onTouchEvent(MotionEvent event) {  
  
    int x = (int) event.getX();  
    int y = (int) event.getY();  
    int action = event.getAction();  
  
    switch (action) {  
        case MotionEvent.ACTION_DOWN:  
Log.i("CustomView", "onTouchEvent: ACTION_DOWN: x = " + x + ", y = " + y);  
        break;  
  
        case MotionEvent.ACTION_MOVE:  
Log.i("CustomView", "onTouchEvent: ACTION_MOVE: x = " + x + ", y = " + y);  
        break;  
  
        case MotionEvent.ACTION_UP:  
Log.i("CustomView", "onTouchEvent: ACTION_UP: x = " + x + ", y = " + y);  
        break;  
    }  
    return true;  
}
```

Further reading:

- [Android official documentation: Responding to Touch Events](#)

第29章：获取计算后的视图尺寸

第29.1节：在Activity中计算初始视图尺寸

```
package com.example;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;

public class ExampleActivity extends Activity {

    @Override
    protected void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        final View viewToMeasure = findViewById(R.id.view_to_measure);

        // 此时尚未知待测视图的尺寸。
        // viewToMeasure.getWidth() 和 viewToMeasure.getHeight() 都返回0,
        // 无论屏幕上的实际大小如何。

        viewToMeasure.getViewTreeObserver().addOnPreDrawListener(new
            ViewTreeObserver.OnPreDrawListener() {
                @Override
                public boolean onPreDraw() {
                    // 待测视图现在已被测量和布局，显示尺寸已知。
                    logComputedViewDimensions(viewToMeasure.getWidth(), viewToMeasure.getHeight());

                    // 移除此监听器，因为我们已经成功计算出所需的尺寸。
                    viewToMeasure.getViewTreeObserver().removeOnPreDrawListener(this);

                    // 始终返回 true 以继续绘制。
                    return true;
                }
            });
    }

    private void logComputedViewDimensions(final int width, final int height) {
        Log.d("example", "viewToMeasure 的宽度为 " + width);
        Log.d("example", "viewToMeasure 的高度为 " + height);
    }
}
```

Chapter 29: Getting Calculated View Dimensions

Section 29.1: Calculating initial View dimensions in an Activity

```
package com.example;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;

public class ExampleActivity extends Activity {

    @Override
    protected void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        final View viewToMeasure = findViewById(R.id.view_to_measure);

        // viewToMeasure dimensions are not known at this point.
        // viewToMeasure.getWidth() and viewToMeasure.getHeight() both return 0,
        // regardless of on-screen size.

        viewToMeasure.getViewTreeObserver().addOnPreDrawListener(new
            ViewTreeObserver.OnPreDrawListener() {
                @Override
                public boolean onPreDraw() {
                    // viewToMeasure is now measured and laid out, and displayed dimensions are known.
                    logComputedViewDimensions(viewToMeasure.getWidth(), viewToMeasure.getHeight());

                    // Remove this listener, as we have now successfully calculated the desired
                    // dimensions.
                    viewToMeasure.getViewTreeObserver().removeOnPreDrawListener(this);

                    // Always return true to continue drawing.
                    return true;
                }
            });
    }

    private void logComputedViewDimensions(final int width, final int height) {
        Log.d("example", "viewToMeasure has width " + width);
        Log.d("example", "viewToMeasure has height " + height);
    }
}
```

第30章：向 Android 项目添加 FuseView

从 fusetools 导出一个 `Fuse.View` 并在现有的 Android 项目中使用它。

我们的目标是导出整个 hikr 示例应用 并在一个 `Activity` 中使用它。

最终作品可在 [@lucamtudor/hikr-fuse-view](#)

第30.1节：hikr 应用，仅仅是另一个 `android.view.View`

先决条件

- 你应该安装了 fuse (<https://www.fusetools.com/downloads>)
- 你应该已经完成了入门教程
- 在终端中：`fuse install android`
- 在终端中：`uno install Fuse.Views`

步骤 1

```
git clone https://github.com/fusetools/hikr
```

步骤 2：添加包引用到 `Fuse.Views`

在项目根目录中找到 `hikr.unoproj` 文件，并将 "`Fuse.Views`" 添加到 "`Packages`" 数组中。

```
{
  "RootNamespace": "",
  "Packages": [
    "Fuse",
    "FuseJS",
    "Fuse.Views"
  ],
  "Includes": [
    "*",
    "Modules/*.js:Bundle"
  ]
}
```

步骤3：制作HikrApp组件以承载整个应用

3.1 在项目根目录下新建一个名为 `HikrApp.ux` 的文件，并粘贴 `MainView.ux` 的内容。

HikrApp.ux

```
<App Background="#022328">
  <iOS.StatusBarConfig Style="Light" />
  <Android.StatusBarConfig Color="#022328" />

  <Router ux:Name="router" />

  <ClientPanel>
    <Navigator DefaultPath="splash">
```

Chapter 30: Adding a FuseView to an Android Project

Export a `Fuse.View` from [fusetools](#) and use it inside an existing android project.

Our goal is to export the entire [hikr sample app](#) and use it inside an `Activity`.

Final work can be found [@lucamtudor/hikr-fuse-view](#)

Section 30.1: hikr app, just another `android.view.View`

Prerequisites

- you should have fuse installed (<https://www.fusetools.com/downloads>)
- you should have done the [introduction tutorial](#)
- in terminal: `fuse install android`
- in terminal: `uno install Fuse.Views`

Step 1

```
git clone https://github.com/fusetools/hikr
```

Step 2 : Add package reference to `Fuse.Views`

Find `hikr.unoproj` file inside the project root folder and add "`Fuse.Views`" to the "`Packages`" array.

```
{
  "RootNamespace": "",
  "Packages": [
    "Fuse",
    "FuseJS",
    "Fuse.Views"
  ],
  "Includes": [
    "*",
    "Modules/*.js:Bundle"
  ]
}
```

Step 3 : Make HikrApp component to hold the entire app

3.1 In the project root folder make a new file called `HikrApp.ux` and paste the contents of `MainView.ux`.

HikrApp.ux

```
<App Background="#022328">
  <iOS.StatusBarConfig Style="Light" />
  <Android.StatusBarConfig Color="#022328" />

  <Router ux:Name="router" />

  <ClientPanel>
    <Navigator DefaultPath="splash">
```

```

<SplashPage ux:Template="splash" router="router" />
<HomePage ux:Template="home" router="router" />
<EditHikePage ux:Template="editHike" router="router" />
</Navigator>
</ClientPanel>
</App>

```

3.2 在HikrApp.ux中

- 将<App>标签替换为<Page>
- 在打开的 <Page> 中添加 ux:Class="HikrApp"
- 移除 <ClientPanel>, 我们不再需要担心状态栏或底部导航按钮

HikrApp.ux

```

<Page ux:Class="HikrApp" Background="#022328">
  <iOS.StatusBarConfig Style="Light" />
  <Android.StatusBarConfig Color="#022328" />

  <Router ux:Name="router" />

  <Navigator DefaultPath="splash">
    <SplashPage ux:Template="splash" router="router" />
    <HomePage ux:Template="home" router="router" />
    <EditHikePage ux:Template="editHike" router="router" />
  </Navigator>
</Page>

```

3.3 在 MainView.ux 中使用新创建的 HikrApp 组件

将 MainView.ux 文件的内容替换为：

```

<App>
  <HikrApp/>
</App>

```

我们的应用恢复了正常的行为，但现在我们已将其提取为一个名为 HikrApp 的独立组件

步骤4 在 MainView.ux 中将 <App> 标签替换为 <ExportedViews> 并添加 ux:Template="HikrAppView" 到 <HikrApp />

```

<ExportedViews>
  <HikrApp ux:Template="HikrAppView" />
</ExportedViews>

```

记住模板 HikrAppView，因为我们需要它从Java中获取对视图的引用。

注意。根据Fuse文档：

ExportedViews 在进行常规 fuse preview 和 uno build 时表现得像 App

不正确。从Fuse Studio预览时会出现此错误：

```

<SplashPage ux:Template="splash" router="router" />
<HomePage ux:Template="home" router="router" />
<EditHikePage ux:Template="editHike" router="router" />
</Navigator>
</ClientPanel>
</App>

```

3.2 In HikrApp.ux

- replace the <App> tags with <Page>
- add ux:Class="HikrApp" to the opening <Page>
- remove <ClientPanel>, we don't have to worry anymore about the status bar or the bottom nav buttons

HikrApp.ux

```

<Page ux:Class="HikrApp" Background="#022328">
  <iOS.StatusBarConfig Style="Light" />
  <Android.StatusBarConfig Color="#022328" />

  <Router ux:Name="router" />

  <Navigator DefaultPath="splash">
    <SplashPage ux:Template="splash" router="router" />
    <HomePage ux:Template="home" router="router" />
    <EditHikePage ux:Template="editHike" router="router" />
  </Navigator>
</Page>

```

3.3 Use the newly created HikrApp component inside MainView.ux

Replace the content of MainView.ux file with:

```

<App>
  <HikrApp/>
</App>

```

Our app is back to its normal behavior, but we now have extracted it to a separate component called HikrApp

Step 4 Inside MainView.ux replace the <App> tags with <ExportedViews> and add ux:Template="HikrAppView" to <HikrApp />

```

<ExportedViews>
  <HikrApp ux:Template="HikrAppView" />
</ExportedViews>

```

Remember the template HikrAppView, because we'll need it to get a reference to our view from Java.

Note. From the fuse docs:

ExportedViews will behave as App when doing normal fuse preview and uno build

Not true. You will get this error when previewing from Fuse Studio:

错误：在任何包含的UX文件中都找不到App标签。你是否忘记包含包含app标签的UX文件？

Error: Couldn't find an App tag in any of the included UX files. Have you forgot to include the UX file that contains the app tag?

步骤5 用 <GraphicsView> 包裹 SplashPage.ux 中的 <DockPanel>

```
<Page ux:Class="SplashPage">
  <Router ux:Dependency="router" />

  <JavaScript File="SplashPage.js" />

  <GraphicsView>
    <DockPanel ClipToBounds="true">
      <Video Layer="Background" File="../Assets/nature.mp4" IsLooping="true" AutoPlay="true"
        StretchMode="UniformToFill" Opacity="0.5">
        <Blur Radius="4.75" />
      </Video>

      <hikr.Text Dock="Bottom" Margin="10" Opacity=".5" TextAlignment="Center"
        FontSize="12">原始视频由Graham Uhelski提供</hikr.Text>

      <Grid RowCount="2">
        <StackPanel Alignment="VerticalCenter">
          <hikr.Text Alignment="HorizontalCenter" FontSize="70">hikr</hikr.Text>
          <hikr.Text Alignment="HorizontalCenter" Opacity=".5">走出去</hikr.Text>
        </StackPanel>

        <hikr.Button Text="开始使用" FontSize="18" Margin="50,0"
          Alignment="VerticalCenter" Clicked="{goToHomePage}" />
      </Grid>
    </DockPanel>
  </GraphicsView>
</Page>
```

步骤6 将fuse项目导出为aar库

- 在终端，根项目文件夹中：uno clean
- 在终端，根项目文件夹中：uno build -t=android -DLIBRARY

步骤7 准备你的安卓项目

- 将aar从.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar复制到.../androidRootProject/app/libs
- 在根目录的build.gradle文件中添加flatDir { dirs 'libs' }

// 顶层构建文件，可在此添加所有子项目/模块通用的配置选项。

```
buildscript { ... }
```

...

```
allprojects {
  repositories {
    jcenter()
```

Step 5 Wrap SplashPage.ux's <DockPanel> in a <GraphicsView>

```
<Page ux:Class="SplashPage">
  <Router ux:Dependency="router" />

  <JavaScript File="SplashPage.js" />

  <GraphicsView>
    <DockPanel ClipToBounds="true">
      <Video Layer="Background" File="../Assets/nature.mp4" IsLooping="true" AutoPlay="true"
        StretchMode="UniformToFill" Opacity="0.5">
        <Blur Radius="4.75" />
      </Video>

      <hikr.Text Dock="Bottom" Margin="10" Opacity=".5" TextAlignment="Center"
        FontSize="12">original video by Graham Uhelski</hikr.Text>

      <Grid RowCount="2">
        <StackPanel Alignment="VerticalCenter">
          <hikr.Text Alignment="HorizontalCenter" FontSize="70">hikr</hikr.Text>
          <hikr.Text Alignment="HorizontalCenter" Opacity=".5">get out there</hikr.Text>
        </StackPanel>

        <hikr.Button Text="Get Started" FontSize="18" Margin="50,0"
          Alignment="VerticalCenter" Clicked="{goToHomePage}" />
      </Grid>
    </DockPanel>
  </GraphicsView>
</Page>
```

Step 6 Export the fuse project as an aar library

- in terminal, in root project folder: uno clean
- in terminal, in root project folder: uno build -t=android -DLIBRARY

Step 7 Prepare your android project

- copy the aar from .../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar to.../androidRootProject/app/libs
- add flatDir { dirs 'libs' } to the root build.gradle file

// Top-level build file where you can add configuration options common to all sub-projects/modules.

```
buildscript { ... }
```

...

```
allprojects {
  repositories {
    jcenter()
```

```

flatDir {
    dirs 'libs'
}
}

...

```

- 在 app/build.gradle 的依赖中添加 compile(name: 'app-debug', ext: 'aar')

```

apply plugin: 'com.android.application'

android {
compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.shiftstudio.fuseviewtest"
        minSdkVersion 16
    targetSdkVersion 25
        versionCode 1
    versionName "1.0"
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}

dependencies {
    compile('name: app-debug', ext: 'aar')
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    testCompile 'junit:junit:4.12'
}

```

- 在 AndroidManifest.xml 中的 activity 内添加以下属性

```

        android:launchMode="singleTask"
        android:taskAffinity=""
        android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize"

```

你的AndroidManifest.xml将如下所示：

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.shiftstudio.fuseviewtest">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"

```

```

flatDir {
    dirs 'libs'
}
}

...

```

- add compile(name: 'app-debug', ext: 'aar') to dependencies in app/build.gradle

```

apply plugin: 'com.android.application'

android {
compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.shiftstudio.fuseviewtest"
        minSdkVersion 16
    targetSdkVersion 25
        versionCode 1
    versionName "1.0"
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}

dependencies {
    compile('name: app-debug', ext: 'aar')
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    testCompile 'junit:junit:4.12'
}

```

- add the following properties to the activity inside AndroidManifest.xml

```

        android:launchMode="singleTask"
        android:taskAffinity=""
        android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize"

```

Your AndroidManifest.xml will look like this:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.shiftstudio.fuseviewtest">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"

```

```

    android:launchMode="singleTask"
    android:taskAffinity=""
    android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

```

步骤 8: 在您的Activity中显示Fuse.View HikrAppView

- 请注意，您的Activity需要继承FuseViewsActivity

```

public class MainActivity extends FuseViewsActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final ViewHandle fuseHandle = ExportedViews.instantiate("HikrAppView");

        final FrameLayout root = (FrameLayout) findViewById(R.id.fuse_root);
        final View fuseApp = fuseHandle.getView();
        root.addView(fuseApp);
    }
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.shiftstudio.fuseviewtest.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_height="wrap_content"
        android:text="Hello World, from Kotlin" />

    <FrameLayout
        android:id="@+id/fuse_root"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

```

```

    android:launchMode="singleTask"
    android:taskAffinity=""
    android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>

```

Step 8: Show the Fuse.View HikrAppView in your Activity

- note that your Activity needs to inherit FuseViewsActivity

```

public class MainActivity extends FuseViewsActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final ViewHandle fuseHandle = ExportedViews.instantiate("HikrAppView");

        final FrameLayout root = (FrameLayout) findViewById(R.id.fuse_root);
        final View fuseApp = fuseHandle.getView();
        root.addView(fuseApp);
    }
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.shiftstudio.fuseviewtest.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_height="wrap_content"
        android:text="Hello World, from Kotlin" />

    <FrameLayout
        android:id="@+id/fuse_root"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

```

```
<TextView  
    android:layout_width="wrap_content" androi  
    d:text="这是来自原生的。在Fuse视图后面" android:layout_gravity="c  
    enter"  
    android:textStyle="bold"  
    android:textSize="30sp"  
    android:background="@color/colorAccent"  
    android:textAlignment="center"  
    android:layout_height="wrap_content" />  
  
</FrameLayout>  
  
</LinearLayout>
```

注意

当你按下返回键时，在安卓上，应用程序会崩溃。你可以在fuse论坛上跟踪该问题。

```
A/libc: Fatal signal 11 (SIGSEGV), code 1, fault addr 0xdeadcab1 in tid 18026 (io.fuseviewtest)  
[ 05-25 11:52:33.658 16567:16567 W/ ]
```

```
debuggerd: 处理请求: pid=18026 uid=10236 gid=10236 tid=18026
```

最终结果大致如下。你也可以在github上找到一个简短的视频片段。

```
<TextView  
    android:layout_width="wrap_content"  
    android:text="THIS IS FROM NATIVE.\nBEHIND FUSE VIEW"  
    android:layout_gravity="center"  
    android:textStyle="bold"  
    android:textSize="30sp"  
    android:background="@color/colorAccent"  
    android:textAlignment="center"  
    android:layout_height="wrap_content" />  
  
</FrameLayout>  
  
</LinearLayout>
```

Note

When you press the back button, on android, the app crashes. You can follow the issue on the [fuse forum](#).

```
A/libc: Fatal signal 11 (SIGSEGV), code 1, fault addr 0xdeadcab1 in tid 18026 (io.fuseviewtest)  
[ 05-25 11:52:33.658 16567:16567 W/ ]  
debuggerd: handling request: pid=18026 uid=10236 gid=10236 tid=18026
```

And the final result is something like this. You can also find a short clip on [github](#).

Fuse View Test

Hello World, from Kotlin

hikr

get out there

Get Started

original video by Graham Uhelski



Fuse View Test

Hello World, from Kotlin

hikr

get out there

Get Started

original video by Graham Uhelski



第31章：支持不同分辨率、尺寸的屏幕

第31.1节：使用配置限定符

Android支持多种配置限定符，允许你根据当前设备屏幕的特性控制系统如何选择你的备用资源。配置限定符是一个字符串，你可以将其附加到Android项目中的资源目录名后，指定该目录内资源所针对的配置。

使用配置限定符的方法：

- 在项目的res/目录下创建一个新目录，命名格式为：`<resources_name>-<qualifier>`。`<resources_name>`是标准资源名称（例如drawable或layout）。
- `<qualifier>`是配置限定符，指定这些资源所针对的屏幕配置。可用于（例如 hdpi 或 xlarge）。

例如，以下应用资源目录为不同屏幕

尺寸和不同的可绘制资源提供了不同的布局设计。启动器图标请使用mipmap/文件夹。

```
res/layout/my_layout.xml           // 适用于普通屏幕尺寸（“默认”）的布局
res/layout-large/my_layout.xml     // 适用于大屏幕尺寸的布局
res/layout-xlarge/my_layout.xml    // 适用于超大屏幕尺寸的布局
res/layout-xlarge-land/my_layout.xml // 适用于横屏方向的超大屏幕布局

res/drawable-mdpi/graphic.png     // 适用于中密度的位图
res/drawable-hdpi/graphic.png     // 适用于高密度的位图
res/drawable-xhdpi/graphic.png    // 适用于超高密度的位图
res/drawable-xxhdpi/graphic.png   // 适用于超超高密度的位图

res/mipmap-mdpi/my_icon.png       // 适用于中密度的启动器图标
res/mipmap-hdpi/my_icon.png       // 适用于高密度的启动器图标
res/mipmap-xhdpi/my_icon.png      // 适用于超高密度的启动器图标
res/mipmap-xxhdpi/my_icon.png     // 适用于超超高密度的启动器图标
res/mipmap-xxxhdpi/my_icon.png    // 适用于超超超高密度的启动器图标
```

第31.2节：将dp和sp转换为像素

当你需要为诸如Paint.setTextSize设置像素值，但仍希望根据设备进行缩放时，可以转换dp和sp值。

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 12f, metrics);

DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 12f, metrics);
```

或者，如果你有上下文来加载资源，也可以将尺寸资源转换为像素。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="size_in_sp">12sp</dimen>
    <dimen name="size_in_dp">12dp</dimen>
</resources>
```

Chapter 31: Supporting Screens With Different Resolutions, Sizes

Section 31.1: Using configuration qualifiers

Android supports several configuration qualifiers that allow you to control how the system selects your alternative resources based on the characteristics of the current device screen. A configuration qualifier is a string that you can append to a resource directory in your Android project and specifies the configuration for which the resources inside are designed.

To use a configuration qualifier:

- Create a new directory in your project's res/ directory and name it using the format: `<resources_name>-<qualifier>`. `<resources_name>` is the standard resource name (such as drawable or layout).
- `<qualifier>` is a configuration qualifier, specifying the screen configuration for which these resources are to be used (such as hdpi or xlarge).

For example, the following application resource directories provide different layout designs for different screen sizes and different drawables. Use the mipmap/ folders for launcher icons.

```
res/layout/my_layout.xml           // layout for normal screen size ("default")
res/layout-large/my_layout.xml     // layout for large screen size
res/layout-xlarge/my_layout.xml    // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra-large in landscape orientation

res/drawable-mdpi/graphic.png     // bitmap for medium-density
res/drawable-hdpi/graphic.png     // bitmap for high-density
res/drawable-xhdpi/graphic.png    // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png   // bitmap for extra-extra-high-density

res/mipmap-mdpi/my_icon.png       // launcher icon for medium-density
res/mipmap-hdpi/my_icon.png       // launcher icon for high-density
res/mipmap-xhdpi/my_icon.png      // launcher icon for extra-high-density
res/mipmap-xxhdpi/my_icon.png     // launcher icon for extra-extra-high-density
res/mipmap-xxxhdpi/my_icon.png    // launcher icon for extra-extra-extra-high-density
```

Section 31.2: Converting dp and sp to pixels

When you need to set a pixel value for something like `Paint.setTextSize` but still want it be scaled based on the device, you can convert dp and sp values.

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 12f, metrics);

DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 12f, metrics);
```

Alternatively, you can convert a dimension resource to pixels if you have a context to load the resource from.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="size_in_sp">12sp</dimen>
    <dimen name="size_in_dp">12dp</dimen>
</resources>
```

```
// 获取资源中指定的精确尺寸  
float pixels = context.getResources().getDimension(R.dimen.size_in_sp);  
float pixels = context.getResources().getDimension(R.dimen.size_in_dp);  
  
// 获取资源中指定的尺寸，用作大小。  
// 该值向下取整到最接近的整数，但至少为1像素。  
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_sp);  
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_dp);  
  
// 获取资源中指定的尺寸，用作偏移量。  
// 该值向下取整到最接近的整数，且可以为0像素。  
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_sp);  
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_dp);
```

第31.3节：文本大小与不同的安卓屏幕尺寸

有时，只有三个选项更好

```
style="@+id/TextAppearance.Small"  
style="@+id/TextAppearance.Medium"  
style="@+id/TextAppearance.Large"
```

使用 small 和 large 来区分正常屏幕尺寸。

```
<TextView  
    android:id="@+id/TextViewTopBarTitle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    style="@+id/TextAppearance.Small" />
```

对于普通情况，您无需指定任何内容。

```
<TextView  
    android:id="@+id/TextViewTopBarTitle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

使用这个，您可以避免为不同屏幕尺寸测试和指定尺寸。

```
// Get the exact dimension specified by the resource  
float pixels = context.getResources().getDimension(R.dimen.size_in_sp);  
float pixels = context.getResources().getDimension(R.dimen.size_in_dp);  
  
// Get the dimension specified by the resource for use as a size.  
// The value is rounded down to the nearest integer but is at least 1px.  
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_sp);  
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_dp);  
  
// Get the dimension specified by the resource for use as an offset.  
// The value is rounded down to the nearest integer and can be 0px.  
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_sp);  
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_dp);
```

Section 31.3: Text size and different android screen sizes

Sometimes, it's better to have only three options

```
style="@+id/TextAppearance.Small"  
style="@+id/TextAppearance.Medium"  
style="@+id/TextAppearance.Large"
```

Use small and large to differentiate from normal screen size.

```
<TextView  
    android:id="@+id/TextViewTopBarTitle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    style="@+id/TextAppearance.Small" />
```

For normal, you don't have to specify anything.

```
<TextView  
    android:id="@+id/TextViewTopBarTitle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

Using this, you can avoid testing and specifying dimensions for different screen sizes.

第32章：ViewFlipper

ViewFlipper 是一个 ViewAnimator，用于在添加到它的两个或多个视图之间切换。一次只显示一个子视图。如果需要，ViewFlipper 可以以固定间隔自动切换每个子视图。

第32.1节：带图片滑动的 ViewFlipper

XML 文件：

```
<ViewFlipper  
    android:id="@+id/viewflip"  
    android:layout_width="match_parent"  
    android:layout_height="250dp"  
    android:layout_weight="1"  
/>
```

Java 代码：

```
public class BlankFragment extends Fragment{  
    ViewFlipper viewFlipper;  
    FragmentManager fragmentManager;  
    int gallery_grid_Images[] = {drawable.image1, drawable.image2, drawable.image3,  
        drawable.image1, drawable.image2, drawable.image3, drawable.image1,  
        drawable.image2, drawable.image3, drawable.image1  
    };  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){  
        View rootView = inflater.inflate(fragment_blank, container, false);  
        viewFlipper = (ViewFlipper)rootView.findViewById(R.id.viewflip);  
        for(int i=0; i<gallery_grid_Images.length; i++){  
            // 这将创建动态的图像视图并添加到 ViewFlipper 中。  
            setFlipperImage(gallery_grid_Images[i]);  
        }  
        return rootView;  
    }  
  
    private void setFlipperImage(int res) {  
        Log.i("Set Filpper Called", res+"");  
        ImageView image = new ImageView(getContext());  
        image.setBackgroundResource(res);  
        viewFlipper.addView(image);  
        viewFlipper.setFlipInterval(1000);  
        viewFlipper.setAutoStart(true);  
    }  
}
```

Chapter 32: ViewFlipper

A ViewFlipper is a ViewAnimator that switches between two or more views that have been added to it. Only one child is shown at a time. If requested, the ViewFlipper can automatically flip between each child at a regular interval.

Section 32.1: ViewFlipper with image sliding

XML file:

```
<ViewFlipper  
    android:id="@+id/viewflip"  
    android:layout_width="match_parent"  
    android:layout_height="250dp"  
    android:layout_weight="1"  
/>
```

Java code:

```
public class BlankFragment extends Fragment{  
    ViewFlipper viewFlipper;  
    FragmentManager fragmentManager;  
    int gallery_grid_Images[] = {drawable.image1, drawable.image2, drawable.image3,  
        drawable.image1, drawable.image2, drawable.image3, drawable.image1,  
        drawable.image2, drawable.image3, drawable.image1  
    };  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){  
        View rootView = inflater.inflate(fragment_blank, container, false);  
        viewFlipper = (ViewFlipper)rootView.findViewById(R.id.viewflip);  
        for(int i=0; i<gallery_grid_Images.length; i++){  
            // This will create dynamic image views and add them to the ViewFlipper.  
            setFlipperImage(gallery_grid_Images[i]);  
        }  
        return rootView;  
    }  
  
    private void setFlipperImage(int res) {  
        Log.i("Set Filpper Called", res+"");  
        ImageView image = new ImageView(getContext());  
        image.setBackgroundResource(res);  
        viewFlipper.addView(image);  
        viewFlipper.setFlipInterval(1000);  
        viewFlipper.setAutoStart(true);  
    }  
}
```

第33章：设计模式

设计模式是程序员在设计应用程序或系统时用来解决常见问题的正式最佳实践。

设计模式通过提供经过测试和验证的开发范例，可以加快开发过程。

重用设计模式有助于防止可能导致重大问题的细微错误，同时也提高了熟悉这些模式的程序员和架构师的代码可读性。

第33.1节：观察者模式

观察者模式是一种常见模式，广泛应用于许多场景。一个真实的例子可以取自YouTube：当你喜欢一个频道并希望获取该频道的所有新闻和观看新视频时，你必须订阅该频道。然后，每当该频道发布任何新闻时，你（以及所有其他订阅者）都会收到通知。

观察者将包含两个组件。一个是广播者（频道），另一个是接收者（你或其他任何订阅者）。广播者将管理所有订阅它的接收者实例。当广播者触发新事件时，它会向所有接收者实例宣布该事件。当接收者收到事件时，它必须对该事件做出反应，例如打开YouTube并播放新视频。

实现观察者模式

- 广播者必须提供允许接收者订阅和取消订阅的方法。当广播者触发事件时，订阅者需要被通知事件已发生：

```
class Channel{  
    private List<Subscriber> subscribers;  
    public void subscribe(Subscriber sub) {  
        // 添加新订阅者。  
    }  
    public void unsubscribe(Subscriber sub) {  
        // 移除订阅者。  
    }  
    public void newEvent() {  
        // 通知所有订阅者的事件。  
    }  
}
```

- 接收者需要实现一个方法来处理来自广播者的事件：

```
interface 订阅者 {  
    void doSubscribe(频道 channel);  
    void doUnsubscribe(频道 channel);  
    void handleEvent(); // 处理新事件。  
}
```

Chapter 33: Design Patterns

Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

Design patterns can speed up the development process by providing tested, proven development paradigms.

Reusing design patterns helps to prevent subtle issues that can cause major problems, and it also improves code readability for coders and architects who are familiar with the patterns.

Section 33.1: Observer pattern

The observer pattern is a common pattern, which is widely used in many contexts. A real example can be taken from YouTube: When you like a channel and want to get all news and watch new videos from this channel, you have to subscribe to that channel. Then, whenever this channel publishes any news, you (and all other subscribers) will receive a notification.

An observer will have two components. One is a broadcaster (channel) and the other is a receiver (you or any other subscriber). The broadcaster will handle all receiver instances that subscribed to it. When the broadcaster fires a new event, it will announce this to all receiver instances. When the receiver receives an event, it will have to react to that event, for example, by turning on YouTube and playing the new video.

Implementing the observer pattern

- 广播者必须提供允许接收者订阅和取消订阅的方法。当广播者触发事件时，订阅者需要被通知事件已发生：

```
class Channel{  
    private List<Subscriber> subscribers;  
    public void subscribe(Subscriber sub) {  
        // Add new subscriber.  
    }  
    public void unsubscribe(Subscriber sub) {  
        // Remove subscriber.  
    }  
    public void newEvent() {  
        // Notification event for all subscribers.  
    }  
}
```

- 接收者需要实现一个方法来处理来自广播者的事件：

```
interface Subscriber {  
    void doSubscribe(Channel channel);  
    void doUnsubscribe(Channel channel);  
    void handleEvent(); // Process the new event.  
}
```

第33.2节：单例类示例

Java 单例模式

实现单例模式有不同的方法，但它们都有以下共同的概念。

Section 33.2: Singleton Class Example

Java Singleton Pattern

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

- 私有构造函数，限制其他类实例化该类。
- 同一类的私有静态变量，作为该类的唯一实例。
- 公共静态方法，返回该类的实例，这是外部世界获取单例类实例的全局访问点。
-

```
/**
* 单例类。
*/
public final class Singleton {

    /**
     * 私有构造函数，防止外部实例化该类。
     */
    private Singleton() {}

    /**
     * 类的静态实例。
     */
    private static final Singleton INSTANCE = new Singleton();

    /**
     * 供用户调用以获取类的实例。
     *
     * @return 单例实例。
     */
    public static Singleton 获取实例() {
        return INSTANCE;
    }
}
```

- Private constructor to restrict instantiation of the class from other classes.
- Private static variable of the same class that is the only instance of the class.
- Public static method that returns the instance of the class, this is the global access point for outer world to get the instance of the singleton class.

```
/**
* Singleton class.
*/
public final class Singleton {

    /**
     * Private constructor so nobody can instantiate the class.
     */
    private Singleton() {}

    /**
     * Static to class instance of the class.
     */
    private static final Singleton INSTANCE = new Singleton();

    /**
     * To be called by user to obtain instance of the class.
     *
     * @return instance of the singleton.
     */
    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```

第34章：活动

参数	详情
意图	可与startActivity一起使用以启动一个活动
数据包	从字符串键映射到各种Parcelable值的映射。
背景	关于应用环境的全局信息接口。

一个活动代表一个带有用户**界面(UI)**的单一屏幕。一个安卓应用可能有多个活动，例如，一个电子邮件应用可以有一个活动列出所有邮件，另一个活动显示邮件内容，还有另一个活动用于撰写新邮件。应用中的所有活动协同工作，创造完美的用户体验。

第34.1节：活动启动模式

启动模式定义任务中新建或已有活动的行为。

可能的启动模式有：

- standard
- singleTop
- singleTask
- singleInstance

应在安卓清单文件的<activity>元素中通过android:launchMode属性定义。

```
<activity
    android:launchMode=["standard" | "singleTop" | "singleTask" | "singleInstance"] />
```

Standard (标准) :

默认值。如果设置此模式，每个新的意图都会创建一个新的活动实例。因此，可能会有多个相同类型的活动。新活动将被放置在任务的顶部。不同安卓版本存在一些差异：如果活动是从另一个应用启动的，在安卓版本 <= 4.4 上，它将被放置在与启动应用相同的任务中，但在 >= 5.0 版本上会创建新的任务。

SingleTop (单顶模式) :

此模式与standard几乎相同。可以创建多个singleTop活动实例。区别在于，如果活动实例已经存在于当前栈的顶部，onNewIntent()将被调用，而不是创建新的实例。

SingleTask (单任务模式) :

使用此启动模式的活动在系统中只能有一个实例。如果不存在该活动的新任务，则会创建新任务。否则，包含该活动的任务将被移到前台，并调用onNewIntent。

单实例：

此模式类似于 singleTask。区别在于持有 singleInstance 活动的任务只能有该活动，不能有其他活动。当 singleInstance 活动创建另一个活动时，将创建新任务来放置该活动。

Chapter 34: Activity

Parameter	Details
Intent	Can be used with startActivity to launch an Activity
Bundle	A mapping from String keys to various Parcelable values.
Context	Interface to global information about an application environment.

An Activity represents a single screen with a user **interface(UI)**. An Android App may have more than one Activity, for example, An email App can have one activity to list all the emails, another activity to show email contents, yet another activity to compose new email. All the activities in an App work together to create perfect user experience.

Section 34.1: Activity launchMode

Launch mode defines the behaviour of new or existing activity in the task.

There are possible launch modes:

- standard
- singleTop
- singleTask
- singleInstance

It should be defined in android manifest in <activity> element as android:launchMode attribute.

```
<activity
    android:launchMode=[ "standard" | "singleTop" | "singleTask" | "singleInstance" ] />
```

Standard:

Default value. If this mode set, new activity will always be created for each new intent. So it's possible to get many activities of same type. New activity will be placed on the top of the task. There is some difference for different android version: if activity is starting from another application, on androids <= 4.4 it will be placed on same task as starter application, but on >= 5.0 new task will be created.

SingleTop:

This mode is almost the same as standard. Many instances of singleTop activity could be created. The difference is, if an instance of activity already exists on the top of the current stack, onNewIntent() will be called instead of creating new instance.

SingleTask:

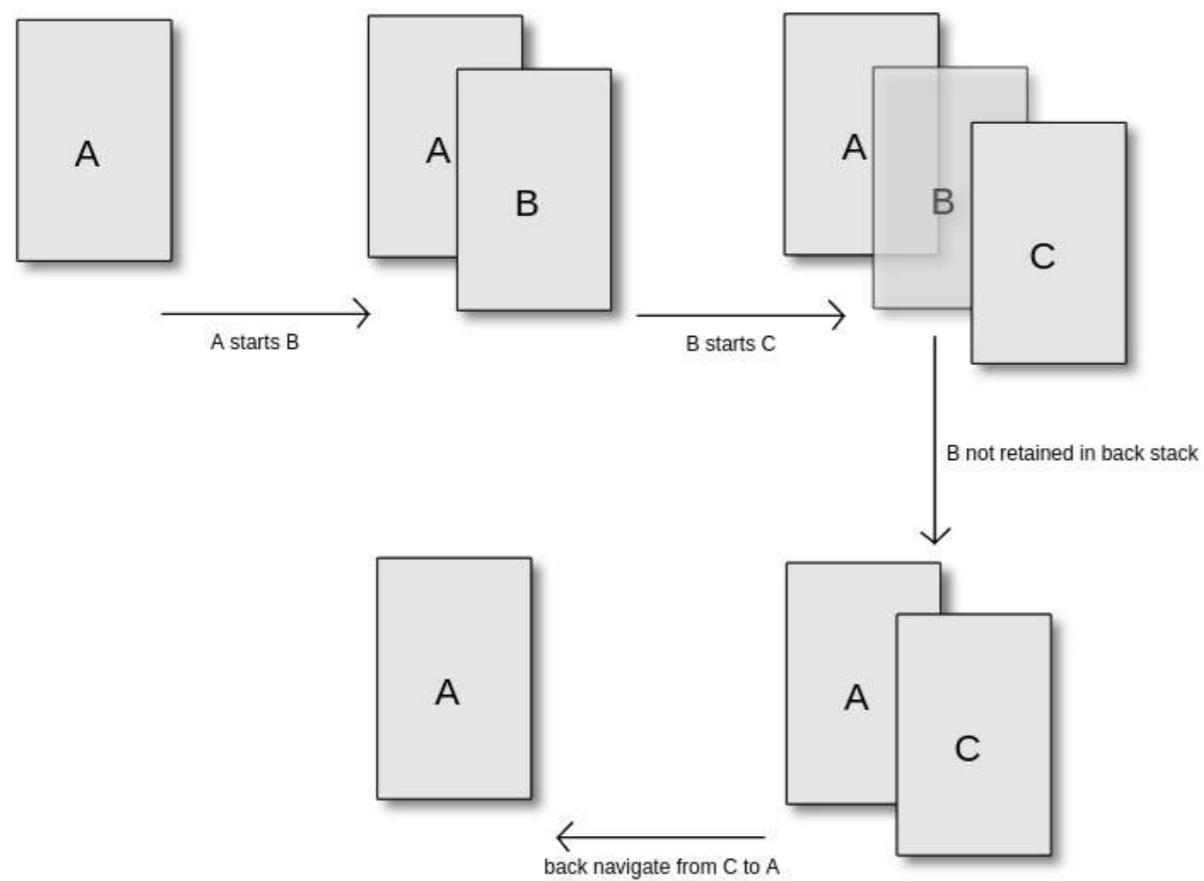
Activity with this launch mode can have only one instance **in the system**. New task for activity will be created, if it doesn't exist. Otherwise, task with activity will be moved to front and onNewIntent will be called.

SingleInstance:

This mode is similar to singleTask. The difference is task that holds an activity with singleInstance could have only this activity and nothing more. When singleInstance activity create another activity, new task will be created to place that activity.

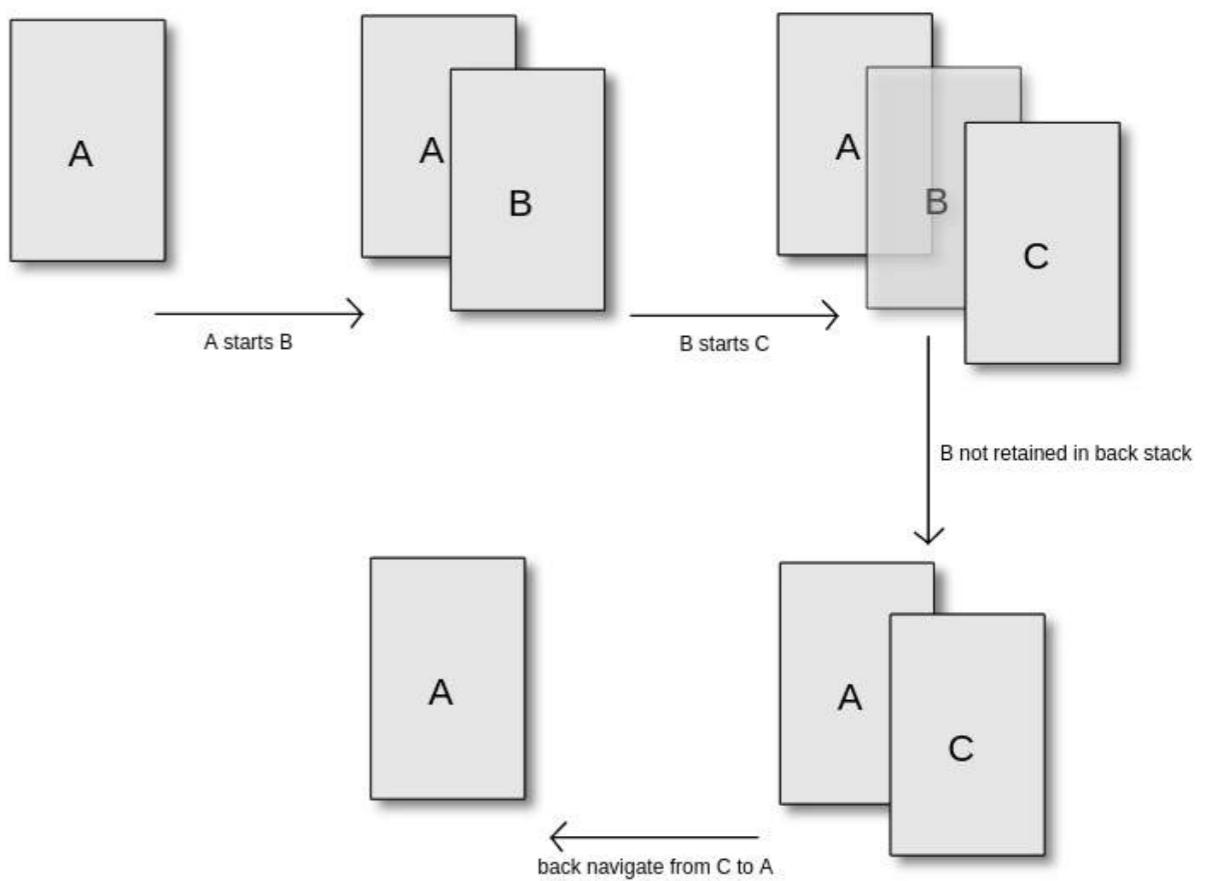
第34.2节：将活动从返回栈历史中排除

假设有一个活动 B 可以被打开，并且可以进一步启动更多活动。但用户在任务活动中向后导航时不应遇到它。



Section 34.2: Exclude an activity from back-stack history

Let there be Activity B that can be opened, and can further start more Activities. But, user should not encounter it when navigating back in task activities.



最简单的解决方案是在 `AndroidManifest.xml` 中对应的 `<activity>` 标签设置属性 `noHistory` 为 `true`：

```
<activity
    android:name=".B"
    android:noHistory="true">
```

如果 B 在启动下一个活动之前调用 `finish()`，也可以实现相同的行为：

```
finish();
startActivity(new Intent(context, C.class));
```

`noHistory` 标志的典型用法是“启动画面”或登录活动。

第34.3节：Android活动生命周期详解

假设有一个应用程序，其中 `MainActivity` 可以通过按钮点击调用下一个 `Activity`。

```
public class MainActivity extends AppCompatActivity {

    private final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

The simplest solution is to set the attribute `noHistory` to `true` for that `<activity>` tag in `AndroidManifest.xml`:

```
<activity
    android:name=".B"
    android:noHistory="true">
```

This same behavior is also possible from code if B calls `finish()` before starting the next activity:

```
finish();
startActivity(new Intent(context, C.class));
```

Typical usage of `noHistory` flag is with "Splash Screen" or Login Activities.

Section 34.3: Android Activity LifeCycle Explained

Assume an application with a `MainActivity` which can call the Next Activity using a button click.

```
public class MainActivity extends AppCompatActivity {

    private final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Log.d(LOG_TAG, "calling onCreate from MainActivity");
}
@Override
protected void onStart() {
    super.onStart();
Log.d(LOG_TAG, "calling onStart from MainActivity");
}
@Override
protected void onResume() {
    super.onResume();
Log.d(LOG_TAG, "calling onResume from MainActivity");
}

@Override
protected void onPause() {
    super.onPause();
Log.d(LOG_TAG, "从 MainActivity 调用 onPause");
}

@Override
protected void onStop() {
    super.onStop();
Log.d(LOG_TAG, "从 MainActivity 调用 onStop");
}

@Override
protected void onDestroy() {
    super.onDestroy();
Log.d(LOG_TAG, "从 MainActivity 调用 onDestroy");
}

@Override
protected void onRestart() {
    super.onRestart();
Log.d(LOG_TAG, "从 MainActivity 调用 onRestart");
}

public void toNextActivity(){
Log.d(LOG_TAG, "调用下一个 Activity");
Intent intent = new Intent(this, NextActivity.class);
startActivity(intent);
} }

```

和

```

public class NextActivity extends AppCompatActivity {
    private final String LOG_TAG = NextActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
        Log.d(LOG_TAG, "调用 Next Activity 的 onCreate");
    }
    @Override
    protected void onStart() {
        super.onStart();
Log.d(LOG_TAG, "调用 Next Activity 的 onStart");
    }
    @Override
    protected void onResume() {

```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Log.d(LOG_TAG, "calling onCreate from MainActivity");
}
@Override
protected void onStart() {
    super.onStart();
    Log.d(LOG_TAG, "calling onStart from MainActivity");
}
@Override
protected void onResume() {
    super.onResume();
    Log.d(LOG_TAG, "calling onResume from MainActivity");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d(LOG_TAG, "calling onPause from MainActivity");
}

@Override
protected void onStop() {
    super.onStop();
    Log.d(LOG_TAG, "calling onStop from MainActivity");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d(LOG_TAG, "calling onDestroy from MainActivity");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.d(LOG_TAG, "calling onRestart from MainActivity");
}

public void toNextActivity(){
    Log.d(LOG_TAG, "calling Next Activity");
    Intent intent = new Intent(this, NextActivity.class);
    startActivity(intent);
} }

```

and

```

public class NextActivity extends AppCompatActivity {
    private final String LOG_TAG = NextActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
        Log.d(LOG_TAG, "calling onCreate from Next Activity");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(LOG_TAG, "calling onStart from Next Activity");
    }
    @Override
    protected void onResume() {

```

```

super.onResume();
Log.d(LOG_TAG, "调用 Next Activity 的 onResume");
}

@Override
protected void onPause() {
    super.onPause();
Log.d(LOG_TAG, "调用 Next Activity 的 onPause");
}

@Override
protected void onStop() {
    super.onStop();
Log.d(LOG_TAG, "调用 Next Activity 的 onStop");
}

@Override
protected void onDestroy() {
    super.onDestroy();
Log.d(LOG_TAG, "调用 Next Activity 的 onDestroy");
}

@Override
protected void onRestart() {
    super.onRestart();
Log.d(LOG_TAG, "调用 Next Activity 的 onRestart");
}
}

```

当应用程序首次创建时

D/MainActivity : 从MainActivity调用onCreate
D/MainActivity : 从MainActivity调用onStart
D/MainActivity : 从MainActivity调用onResume
被调用

当屏幕休眠时

08:11:03.142 D/MainActivity : 从MainActivity调用onPause
08:11:03.192 D/MainActivity : 从MainActivity调用onStop
被调用。并且当它唤醒时再次调用
08:11:55.922 D/MainActivity : 从MainActivity调用onRestart
08:11:55.962 D/MainActivity : 从MainActivity调用onStart
08:11:55.962 D/MainActivity : 从MainActivity调用onResume
被调用

案例1：当从Main Activity调用下一个Activity时

D/MainActivity : 调用下一个Activity
D/MainActivity : 从MainActivity调用onPause
D/NextActivity : 从Next Activity调用onCreate
D/NextActivity : 从Next Activity调用onStart
D/NextActivity : 从Next Activity调用onResume
D/MainActivity : 从MainActivity调用onStop

当使用返回按钮从Next Activity返回Main Activity时

D/NextActivity : 从Next Activity调用onPause
D/MainActivity : 从MainActivity调用onRestart
D/MainActivity : 从MainActivity调用onStart
D/MainActivity : 从MainActivity调用onResume

```

super.onResume();
Log.d(LOG_TAG, "calling onResume from Next Activity");
}

@Override
protected void onPause() {
    super.onPause();
Log.d(LOG_TAG, "calling onPause from Next Activity");
}

@Override
protected void onStop() {
    super.onStop();
Log.d(LOG_TAG, "calling onStop from Next Activity");
}

@Override
protected void onDestroy() {
    super.onDestroy();
Log.d(LOG_TAG, "calling onDestroy from Next Activity");
}

@Override
protected void onRestart() {
    super.onRestart();
Log.d(LOG_TAG, "calling onRestart from Next Activity");
}
}

```

When app is first created

D/MainActivity: calling onCreate from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity
are called

When screen sleeps

08:11:03.142 D/MainActivity: calling onPause from MainActivity
08:11:03.192 D/MainActivity: calling onStop from MainActivity
are called. And again when it wakes up
08:11:55.922 D/MainActivity: calling onRestart from MainActivity
08:11:55.962 D/MainActivity: calling onStart from MainActivity
08:11:55.962 D/MainActivity: calling onResume from MainActivity
are called

Case1: When Next Activity is called from Main Activity

D/MainActivity: calling Next Activity
D/MainActivity: calling onPause from MainActivity
D/NextActivity: calling onCreate from Next Activity
D/NextActivity: calling onStart from Next Activity
D/NextActivity: calling onResume from Next Activity
D/MainActivity: calling onStop from MainActivity

When Returning back to the Main Activity from Next Activity using back button

D/NextActivity: calling onPause from Next Activity
D/MainActivity: calling onRestart from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity

D/NextActivity : 从Next Activity调用onStop
D/NextActivity : 从Next Activity调用onDestroy

案例2：当Activity部分被遮挡（按下概览按钮时）或当应用进入后台且另一个应用完全遮挡它时

D/MainActivity : 从MainActivity调用onPause
D/MainActivity : 从MainActivity调用onStop
当应用回到前台准备接受用户输入时，
D/MainActivity : 从MainActivity调用onRestart
D/MainActivity : 从MainActivity调用onStart
D/MainActivity : 从MainActivity调用onResume
已调用

案例3：当一个活动被调用以满足隐式意图且用户已做出选择。例如，当按下分享按钮且用户必须从显示的应用列表中选择一个应用时

D/MainActivity : 从MainActivity调用onPause

该活动现在可见但不活跃。当选择完成且应用处于活跃状态时

D/MainActivity : 从MainActivity调用onResume
已调用

案例4：
当应用在后台被杀死（以释放资源给另一个前台应用）时，onPause（针对蜂巢版本之前的设备）或 onStop（针对蜂巢版本及之后的设备）将是应用终止前最后被调用的方法。

onCreate和onDestroy在应用每次运行时最多各调用一次。但onPause、onStop、onRestart、onStart、onResume可能在生命周期中被调用多次。

第34.4节：使用排除最近任务结束应用

首先在AndroidManifest.xml中定义一个ExitActivity

```
<activity
    android:name="com.your_example_app.activities.ExitActivity"
    android:autoRemoveFromRecents="true"
    android:theme="@android:style/Theme.NoDisplay" />
```

随后是 ExitActivity 类

```
/**
 * 用于退出应用程序且不保留在最近打开的应用堆栈中
 */
public class ExitActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (Utils.hasLollipop()) {
            finishAndRemoveTask();
        } else if (Utils.hasJellyBean()) {
            finishAffinity();
        } else {
            finish();
        }
    }
}
```

D/NextActivity: calling onStop from Next Activity
D/NextActivity: calling onDestroy from Next Activity

Case2: When Activity is partially obscured (When overview button is pressed) or When app goes to background and another app completely obscures it

D/MainActivity: calling onPause from MainActivity
D/MainActivity: calling onStop from MainActivity
and when the app is back in the foreground ready to accept User inputs,
D/MainActivity: calling onRestart from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity
are called

Case3: When an activity is called to fulfill implicit intent and user has make a selection. For eg., when share button is pressed and user has to select an app from the list of applications shown

D/MainActivity: calling onPause from MainActivity

The activity is visible but not active now. When the selection is done and app is active
D/MainActivity: calling onResume from MainActivity
is called

Case4:

When the app is killed in the background(to free resources for another foreground app), onPause(for pre-honeycomb device) or onStop(for since honeycomb device) will be the last to be called before the app is terminated.

onCreate and onDestroy will be called utmost once each time the application is run. But the onPause, onStop, onRestart, onStart, onResume maybe called many times during the lifecycle.

Section 34.4: End Application with exclude from Recents

First define an ExitActivity in the AndroidManifest.xml

```
<activity
    android:name="com.your_example_app.activities.ExitActivity"
    android:autoRemoveFromRecents="true"
    android:theme="@android:style/Theme.NoDisplay" />
```

Afterwards the ExitActivity-class

```
/**
 * Activity to exit Application without staying in the stack of last opened applications
 */
public class ExitActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (Utils.hasLollipop()) {
            finishAndRemoveTask();
        } else if (Utils.hasJellyBean()) {
            finishAffinity();
        } else {
            finish();
        }
    }
}
```

```

    /**
 * 退出应用程序并从最近任务中排除
 *
 * @param context 使用的上下文
 */
public static void exitApplication(ApplicationContext context) {
    Intent intent = new Intent(context, ExitActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NO_ANIMATION | Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
context.startActivity(intent);
}

```

第34.5节：使用 setContentView 展示用户界面

Activity 类负责为你创建一个窗口，你可以通过setContentView将你的用户界面放置其中。

有三种setContentView方法：

- `setContentView(int layoutResID)` - 从布局资源设置活动内容。
- `setContentView(View view)` - 将活动内容设置为一个明确的视图。
- `setContentView(View view, ViewGroup.LayoutParams params)` - 将活动内容设置为一个带有指定参数的明确视图。

当调用setContentView时，该视图会直接放入活动的视图层级中。它本身可以是一个复杂的视图层级。

示例

从资源文件设置内容：

添加资源文件（此例中为 `main.xml`）及视图层级：

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello" />

</FrameLayout>

```

将其设置为活动中的内容：

```

public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 资源将被加载,
        // 将所有顶层视图添加到活动中。
        setContentView(R.layout.main);
    }
}

```

将内容设置为指定视图：

```

    /**
     * Exit Application and Exclude from Recents
     *
     * @param context Context to use
     */
    public static void exitApplication(ApplicationContext context) {
        Intent intent = new Intent(context, ExitActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NO_ANIMATION | Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
        context.startActivity(intent);
    }
}

```

Section 34.5: Presenting UI with setContentView

Activity class takes care of creating a window for you in which you can place your UI with `setContentView`.

There are three `setContentView` methods:

- `setContentView(int layoutResID)` - Set the activity content from a layout resource.
- `setContentView(View view)` - Set the activity content to an explicit view.
- `setContentView(View view, ViewGroup.LayoutParams params)` - Set the activity content to an explicit view with provided params.

When `setContentView` is called, this view is placed directly into the activity's view hierarchy. It can itself be a complex view hierarchy.

Examples

Set content from resource file:

Add resource file (`main.xml` in this example) with view hierarchy:

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello" />

</FrameLayout>

```

Set it as content in activity:

```

public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // The resource will be inflated,
        // adding all top-level views to the activity.
        setContentView(R.layout.main);
    }
}

```

Set content to an explicit view:

```

public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 创建带容器的视图
        final FrameLayout root = new FrameLayout(this);
        final TextView text = new TextView(this);
        text.setText("Hello");
        root.addView(text);

        // 将容器设置为内容视图
        setContentView(root);
    }
}

```

第34.6节：活动的向上导航

在Android中，向上导航是通过在Manifest.xml的activity标签中添加android:parentActivityName=""来实现的。基本上，通过这个标签你告诉系统某个活动的父活动。

这是如何实现的？

```

<uses-permission android:name="android.permission.INTERNET" />

<application
    android:name=".SkillSchoolApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".ui.activities.SplashActivity"
        android:theme="@style/SplashTheme">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ui.activities.MainActivity" />
    <activity android:name=".ui.activities.HomeActivity"
        android:parentActivityName=".ui.activities.MainActivity" /> // 这里我只是告诉系统
MainActivityResult是HomeActivity的父活动

```

现在，当我点击 HomeActivity 工具栏内的箭头时，它会带我返回到父活动。

Java 代码

这里我将编写实现此功能所需的相应 Java 代码。

```

public class HomeActivity extends AppCompatActivity {
    @BindView(R.id.toolbar)
    Toolbar toolbar;

```

```

public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Creating view with container
        final FrameLayout root = new FrameLayout(this);
        final TextView text = new TextView(this);
        text.setText("Hello");
        root.addView(text);

        // Set container as content view
        setContentView(root);
    }
}

```

Section 34.6: Up Navigation for Activities

Up navigation is done in android by adding android:parentActivityName="" in Manifest.xml to the activity tag. Basically with this tag you tell the system about the parent activity of a activity.

How is it done?

```

<uses-permission android:name="android.permission.INTERNET" />

<application
    android:name=".SkillSchoolApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".ui.activities.SplashActivity"
        android:theme="@style/SplashTheme">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ui.activities.MainActivity" />
    <activity android:name=".ui.activities.HomeActivity"
        android:parentActivityName=".ui.activities.MainActivity" /> // HERE I JUST TOLD THE SYSTEM THAT
MainActivity is the parent of HomeActivity
</application>

```

Now when I will click on the arrow inside the toolbar of HomeActivity it will take me back to the parent activity.

Java Code

Here I will write the appropriate Java code required for this functionality.

```

public class HomeActivity extends AppCompatActivity {
    @BindView(R.id.toolbar)
    Toolbar toolbar;

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    ButterKnife.bind(this);
    //由于我使用了自定义工具栏，因此将该工具栏的引用设置给 ActionBar。如果你没有使用自定义工具栏，可以直接跳过这
步，继续下一行
    setSupportActionBar(toolbar);
    getSupportActionBar.setDisplayHomeAsUpEnabled(true); // 这将在工具栏中显示返回箭头。
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    ButterKnife.bind(this);
    //Since i am using custom tool bar i am setting reference of that toolbar to Actionbar. If you
are not using custom then you can simple leave this and move to next line
    setSupportActionBar(toolbar);
    getSupportActionBar.setDisplayHomeAsUpEnabled(true); // this will show the back arrow in
the tool bar.
}
}

```

If you run this code you will see when you press back button it will take you back to MainActivity. For further understanding of Up Navigation i would recommend reading [docs](#)

You can more customize this behaviour upon your needs by overriding

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // Respond to the action bar's Up/Home button
        case android.R.id.home:
            NavUtils.navigateUpFromSameTask(this); // Here you will write your logic for handling up
navigation
            return true;
    }
    return super.onOptionsItemSelected(item);
}

```

Simple Hack

This is simple hack which is mostly used to navigate to parent activity if parent is in backstack. By calling `onBackPressed()` if id is equal to `android.R.id.home`

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id) {
        case android.R.id.home:
            onBackPressed();
            return true;
    }
    return super.onOptionsItemSelected(item);
}

```

Section 34.7: Clear your current Activity stack and launch a new Activity

If you want to clear your current Activity stack and launch a new Activity (for example, logging out of the app and launching a log in Activity), there appears to be two approaches.

1. Target (API >= 16)

Calling `finishAffinity()` from an Activity

2. Target (11 <= API < 16)

```
Intent intent = new Intent(this, LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK
| Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
finish();
```

```
Intent intent = new Intent(this, LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK
| Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
finish();
```

第35章：活动识别

活动识别是检测用户的身体活动，以便在设备上执行某些操作，例如在检测到驾驶时计分，手机静止时关闭WiFi，或用户行走时将铃声音量调至最大。

第35.1节：Google Play ActivityRecognitionAPI

这是一个如何使用Google Play服务的ActivityRecognitionApi的简单示例。虽然这是一个很棒的库，但它不适用于未安装Google Play服务的设备。

[ActivityRecognition API文档](#)

清单文件

```
<!-- 使用活动识别需要此项！ -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient apiClient;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        apiClient = new GoogleApiClient.Builder(this)
            .addApi(ActivityRecognition.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();

        //这只是从ActivityReceiver类获取活动意图
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
```

Chapter 35: Activity Recognition

Activity recognition is the detection of a user's physical activity in order to perform certain actions on the device, such as taking points when a drive is detected, turn wifi off when a phone is still, or putting the ring volume to max when the user is walking.

Section 35.1: Google Play ActivityRecognitionAPI

This is a just a simple example of how to use GooglePlay Service's ActivityRecognitionApi. Although this is a great library, it does not work on devices that do not have Google Play Services installed.

[Docs for ActivityRecognition API](#)

Manifest

```
<!-- This is needed to use Activity Recognition! -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient apiClient;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        apiClient = new GoogleApiClient.Builder(this)
            .addApi(ActivityRecognition.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
```

```

localActivityReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ActivityRecognitionResult recognitionResult =
        ActivityRecognitionResult.extractResult(intent);
        TextView textView = (TextView) findViewById(R.id.activityText);

        //这只是为了获取活动名称。使用风险自负。
    }

    textView.setText(DetectedActivity.zzkf(recognitionResult.getMostProbableActivity().getType()));
}

@Override
protected void onResume() {
    super.onResume();

    //注册本地广播接收器
    localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

    //连接谷歌API客户端
    apiClient.connect();
}

@Override
protected void onPause() {
    super.onPause();

    //取消注册活动识别
    ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(apiClient,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));

    //断开api客户端连接
    apiClient.disconnect();

    //注销本地接收器
    localBroadcastManager.unregisterReceiver(localActivityReceiver);
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    //仅当google api客户端已连接时才注册活动识别
    ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient, 0,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));
}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
}
}

```

ActivityReceiver

```

localActivityReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ActivityRecognitionResult recognitionResult =
        ActivityRecognitionResult.extractResult(intent);
        TextView textView = (TextView) findViewById(R.id.activityText);

        //This is just to get the activity name. Use at your own risk.

        textView.setText(DetectedActivity.zzkf(recognitionResult.getMostProbableActivity().getType()));
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //Connect google api client
        apiClient.connect();
    }

    @Override
    protected void onPause() {
        super.onPause();

        //Unregister for activity recognition
        ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(apiClient,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));

        //Disconnects api client
        apiClient.disconnect();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Only register for activity recognition if google api client has connected
        ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient, 0,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));
    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    }
}

```

ActivityReceiver

```

public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}

```

第35.2节：PathSense 活动识别

[PathSense](#)活动识别是另一个适用于没有Google Play服务设备的优秀库，因为他们构建了自己的活动识别模型，但需要开发者在 <http://developer.pathsense.com> 注册以获取API密钥和客户端ID。

清单文件

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />

    <!-- 你需要从他们的网站 (http://developer.pathsense.com) 获取这些 -->
    <meta-data
        android:name="com.pathsense.android.sdk.CLIENT_ID"
        android:value="YOUR_CLIENT_ID" />
    <meta-data
        android:name="com.pathsense.android.sdk.API_KEY"
        android:value="YOUR_API_KEY" />

```

MainActivity.java

```

public class MainActivity extends AppCompatActivity {

    private PathsenseLocationProviderApi pathsenseLocationProviderApi;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pathsenseLocationProviderApi = PathsenseLocationProviderApi.getInstance(this);

        //这只是从ActivityReceiver类获取活动意图
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
    }
}

```

```

public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}

```

Section 35.2: PathSense Activity Recognition

[PathSense](#) activity recognition is another good library for devices which don't have Google Play Services, as they have built their own activity recognition model, but requires developers register at <http://developer.pathsense.com> to get an API key and Client ID.

Manifest

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />

    <!-- You need to acquire these from their website (http://developer.pathsense.com) -->
    <meta-data
        android:name="com.pathsense.android.sdk.CLIENT_ID"
        android:value="YOUR_CLIENT_ID" />
    <meta-data
        android:name="com.pathsense.android.sdk.API_KEY"
        android:value="YOUR_API_KEY" />

```

MainActivity.java

```

public class MainActivity extends AppCompatActivity {

    private PathsenseLocationProviderApi pathsenseLocationProviderApi;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pathsenseLocationProviderApi = PathsenseLocationProviderApi.getInstance(this);

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
    }
}

```

```

localActivityReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //检测到的Activities对象作为可序列化对象传递
        PathsenseDetectedActivities detectedActivities = (PathsenseDetectedActivities)
intent.getSerializableExtra("ps");
        TextView textView = (TextView) findViewById(R.id.activityText);

        textView.setText(detectedActivities.getMostProbableActivity().getDetectedActivity().name());
    }
}

@Override
protected void onResume() {
    super.onResume();

    //注册本地广播接收器
    localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

    //每次接收到时都会更新，即使与上次更新相同
    pathsenseLocationProviderApi.requestActivityUpdates(ActivityReceiver.class);

//    // 仅在状态变化时更新（例如从ON FOOT变为IN VEHICLE）
//    pathsenseLocationProviderApi.requestActivityChanges(ActivityReceiver.class);
}

@Override
protected void onPause() {
    super.onPause();

pathsenseLocationProviderApi.removeActivityUpdates();

//    pathsenseLocationProviderApi.removeActivityChanges();

    //注销本地接收器
    localBroadcastManager.unregisterReceiver(localActivityReceiver);
}
}

```

ActivityReceiver.java

```

// 你不必使用他们的广播接收器，但最好使用，并根据需要将结果传递给其他类。

public class ActivityReceiver extends PathsenseActivityRecognitionReceiver {

    @Override
    protected void onDetectedActivities(Context context, PathsenseDetectedActivities
pathsenseDetectedActivities) {
        Intent intent = new Intent("activity").putExtra("ps", pathsenseDetectedActivities);
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
    }
}

```

```

localActivityReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //The detectedActivities object is passed as a serializable
        PathsenseDetectedActivities detectedActivities = (PathsenseDetectedActivities)
intent.getSerializableExtra("ps");
        TextView textView = (TextView) findViewById(R.id.activityText);

        textView.setText(detectedActivities.getMostProbableActivity().getDetectedActivity().name());
    }
}

@Override
protected void onResume() {
    super.onResume();

    //Register local broadcast receiver
    localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

    //This gives an update every time it receives one, even if it was the same as the last update
    pathsenseLocationProviderApi.requestActivityUpdates(ActivityReceiver.class);

//    // This gives updates only when it changes (ON FOOT -> IN VEHICLE for example)
//    pathsenseLocationProviderApi.requestActivityChanges(ActivityReceiver.class);

}

@Override
protected void onPause() {
    super.onPause();

    pathsenseLocationProviderApi.removeActivityUpdates();

//    pathsenseLocationProviderApi.removeActivityChanges();

    //Unregister local receiver
    localBroadcastManager.unregisterReceiver(localActivityReceiver);
}
}

```

ActivityReceiver.java

```

// You don't have to use their broadcastreceiver, but it's best to do so, and just pass the result
// as needed to another class.

public class ActivityReceiver extends PathsenseActivityRecognitionReceiver {

    @Override
    protected void onDetectedActivities(Context context, PathsenseDetectedActivities
pathsenseDetectedActivities) {
        Intent intent = new Intent("activity").putExtra("ps", pathsenseDetectedActivities);
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
    }
}

```

第36章：分屏 / 多屏活动

第36.1节：Android Nougat中引入的分屏实现

在清单文件的<activity>或<application>元素中设置此属性以启用或禁用多窗口显示：

```
android:resizeableActivity=["true" | "false"]
```

如果此属性设置为true，则该活动可以在分屏和自由形式模式下启动。如果属性设置为false，则该活动不支持多窗口模式。如果该值为false，且用户尝试在多窗口模式下启动该活动，则该活动将占据全屏。

如果您的应用目标API级别为24，但未为此属性指定值，则该属性的默认值为true。

下面的代码展示了如何指定活动在自由形式模式下显示时的默认大小和位置，以及其最小大小：

```
<--这些是谷歌建议的默认值。-->
<activity android:name=".MyActivity">
<layout android:defaultHeight="500dp"
    android:defaultWidth="600dp"
    android:gravity="top|end"
    android:minHeight="450dp"
    android:minWidth="300dp" />
</activity>
```

多窗口模式下禁用的功能

当设备处于多窗口模式时，某些功能会被禁用或忽略，因为它们对于可能与其他活动或应用共享设备屏幕的活动来说没有意义。此类功能包括：

- 一些系统用户界面自定义选项被禁用；例如，如果应用未以全屏模式运行，则无法隐藏状态栏。
- 系统会忽略对android:screenOrientation属性的更改。

如果您的应用目标API级别为23或更低

如果您的应用目标API级别为23或更低，且用户尝试在多窗口模式下使用该应用，系统会强制调整应用大小，除非应用声明了固定方向。

如果您的应用未声明固定方向，您应在运行Android 7.0或更高版本的设备上启动应用，并尝试将应用置于分屏模式。确认在应用被强制调整大小时用户体验是否可接受。

如果应用声明了固定方向，您应尝试将应用置于多窗口模式。确认此时应用仍保持全屏模式。

Chapter 36: Split Screen / Multi-Screen Activities

Section 36.1: Split Screen introduced in Android Nougat implemented

Set this attribute in your manifest's or element to enable or disable multi-window display:

```
android:resizeableActivity=[ "true" | "false" ]
```

If this attribute is set to true, the activity can be launched in split-screen and freeform modes. If the attribute is set to false, the activity does not support multi-window mode. If this value is false, and the user attempts to launch the activity in multi-window mode, the activity takes over the full screen.

If your app targets API level 24, but you do not specify a value for this attribute, the attribute's value defaults to true.

The following code shows how to specify an activity's default size and location, and its minimum size, when the activity is displayed in freeform mode:

```
<--These are default values suggested by google.-->
<activity android:name=".MyActivity">
<layout android:defaultHeight="500dp"
    android:defaultWidth="600dp"
    android:gravity="top|end"
    android:minHeight="450dp"
    android:minWidth="300dp" />
</activity>
```

Disabled features in multi-window mode

Certain features are disabled or ignored when a device is in multi-window mode, because they don't make sense for an activity which may be sharing the device screen with other activities or apps. Such features include:

- Some System UI customization options are disabled; for example, apps cannot hide the status bar if they are not running in full-screen mode.
- The system ignores changes to the **android:screenOrientation** attribute.

If your app targets API level 23 or lower

If your app targets API level 23 or lower and the user attempts to use the app in multi-window mode, the system forcibly resizes the app unless the app declares a fixed orientation.

If your app does not declare a fixed orientation, you should launch your app on a device running Android 7.0 or higher and attempt to put the app in split-screen mode. Verify that the user experience is acceptable when the app is forcibly resized.

If the app declares a fixed orientation, you should attempt to put the app in multi-window mode. Verify that when you do so, the app remains in full-screen mode.

第37章：材质设计

材质设计是跨平台和设备的视觉、动画及交互设计的综合指南。

第37.1节：添加工具栏

工具栏（Toolbar）是操作栏（ActionBar）的一个泛化，用于应用布局中。虽然操作栏（ActionBar）传统上是由框架控制的活动（Activity）不透明窗口装饰的一部分，但工具栏（Toolbar）可以放置在视图层次结构中的任意嵌套层级。可以通过执行以下步骤添加：

- 确保在您的模块（例如应用）的build.gradle文件中添加以下依赖项
依赖项：

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

- 将您的应用主题设置为不包含ActionBar的主题。为此，请编辑您的styles.xml文件在res/values下，设置一个Theme.AppCompat主题。
在此示例中，我们使用Theme.AppCompat.NoActionBar作为您的AppTheme的父主题：

```
<style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primaryDark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

您也可以使用Theme.AppCompat.Light.NoActionBar或Theme.AppCompat.DayNight.NoActionBar，或任何其他本身不包含ActionBar的主题

- 将工具栏添加到您的活动布局中：

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp" />
```

在工具栏下面，您可以添加其余的布局内容。

- 在您的Activity中，将工具栏设置为该Activity的操作栏（ActionBar）。前提是您正在使用appcompat库和AppCompatActivity，您可以使用setSupportActionBar()方法：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

Chapter 37: Material Design

Material Design is a comprehensive guide for visual, motion, and interaction design across platforms and devices.

Section 37.1: Adding a Toolbar

A Toolbar is a generalization of ActionBar for use within application layouts. While an ActionBar is traditionally part of an Activity's opaque window decor controlled by the framework, a Toolbar may be placed at any arbitrary level of nesting within a view hierarchy. It can be added by performing the following steps:

- Make sure the following dependency is added to your module's (e.g. app's) **build.gradle** file under dependencies:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

- Set the theme for your app to one that does **not** have an ActionBar. To do that, edit your **styles.xml** file under **res/values**, and set a **Theme.AppCompat** theme.
In this example we are using **Theme.AppCompat.NoActionBar** as parent of your AppTheme:

```
<style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primaryDark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

You can also use **Theme.AppCompat.Light.NoActionBar** or **Theme.AppCompat.DayNight.NoActionBar**, or any other theme that does not inherently have an ActionBar

- Add the Toolbar to your activity layout:

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp" />
```

Below the Toolbar you can add the rest of your layout.

- In your Activity, set the Toolbar as the ActionBar for this Activity. Provided that you're using the [appcompat library](#) and an AppCompatActivity, you would use the `setSupportActionBar()` method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

```
}
```

完成上述步骤后，您可以使用getSupportActionBar()方法来操作被设置为操作栏（ActionBar）的工具栏。

例如，您可以如下设置标题：

```
getSupportActionBar().setTitle("Activity Title");
```

例如，您还可以按如下方式设置标题和背景颜色：

```
CharSequence title = "您的应用名称";
SpannableString s = new SpannableString(title);
s.setSpan(new ForegroundColorSpan(Color.RED), 0, title.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);
getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.argb(128, 0, 0, 0)));
```

第37.2节：使用Material Design样式的按钮

AppCompat支持库定义了几种有用的按钮样式，每种样式都继承自一个基础
Widget.AppCompat.Button样式，如果您使用的是AppCompat主题，则默认应用于所有按钮。该
样式有助于确保所有按钮默认外观一致，符合Material Design规范。

在此情况下，强调色为粉色。

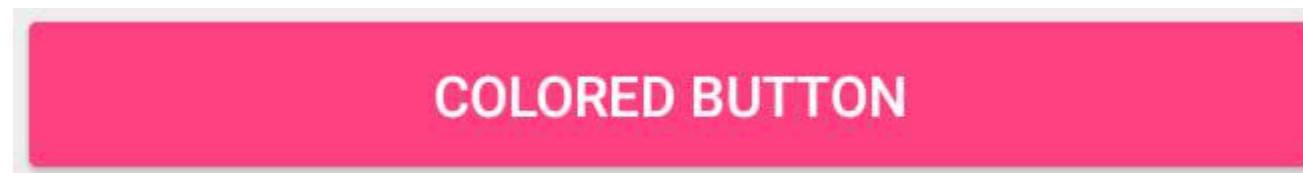
1. 简单按钮：@style/Widget.AppCompat.Button



```
<Button
    style="@style/Widget.AppCompat.Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/simple_button"/>
```

2. 彩色按钮：@style/Widget.AppCompat.Button.Colored

Widget.AppCompat.Button.Colored样式继承自Widget.AppCompat.Button样式，并自动应用您在应用主题中选择的
强调色。



```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="match_parent"
```

```
}
```

After performing the above steps, you can use the getSupportActionBar() method to manipulate the Toolbar that
is set as the ActionBar.

For example, you can set the title as shown below:

```
getSupportActionBar().setTitle("Activity Title");
```

For example, you can also set title and background color as shown below:

```
CharSequence title = "Your App Name";
SpannableString s = new SpannableString(title);
s.setSpan(new ForegroundColorSpan(Color.RED), 0, title.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);
getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.argb(128, 0, 0, 0)));
```

Section 37.2: Buttons styled with Material Design

The [AppCompat Support Library](#) defines several useful styles for [Buttons](#), each of which extend a base
Widget.AppCompat.Button style that is applied to all buttons by default if you are using an AppCompat theme. This
style helps ensure that all buttons look the same by default following the [Material Design specification](#).

In this case the accent color is pink.

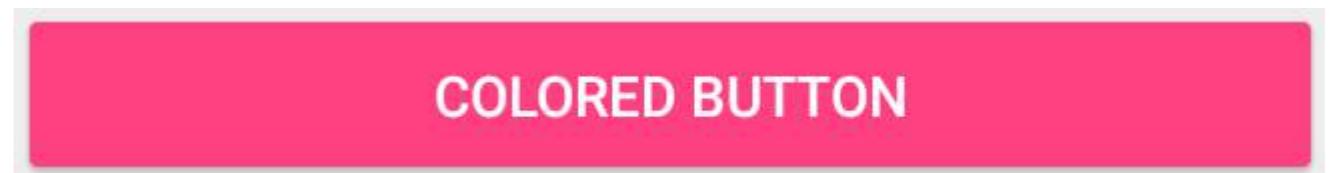
1. Simple Button: @style/Widget.AppCompat.Button



```
<Button
    style="@style/Widget.AppCompat.Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/simple_button"/>
```

2. Colored Button: @style/Widget.AppCompat.Button.Colored

The Widget.AppCompat.Button.Colored style extends the Widget.AppCompat.Button style and applies
automatically the **accent color** you selected in your app theme.



```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/colored_button"/>
```

如果您想自定义背景颜色而不更改主主题中的强调色，可以创建一个自定义主题（继承ThemeOverlay主题）用于您的Button，并将其分配给按钮的`android:theme`属性：

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:theme="@style/MyButtonTheme"/>
```

在res/values/themes.xml中定义主题：

```
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

3. 无边框按钮：@style/Widget.AppCompat.Button.Borderless

BORDERLESS BUTTON

```
<Button
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_button"/>
```

4. 无边框彩色按钮：@style/Widget.AppCompat.Button.Borderless.Colored

BORDERLESS COLORED BUTTON

```
<Button
    style="@style/Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_colored_button"/>
```

第37.3节：添加浮动操作按钮（FAB）

在材质设计中，浮动操作按钮代表一个活动中的主要操作。它们的特点是在用户界面上方悬浮着一个带圈的图标，并具有包括变形在内的运动行为，

```
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/colored_button"/>
```

If you want to customize the background color without changing the accent color in your *main theme* you can create a *custom theme* (extending the ThemeOverlay theme) for your `Button` and assign it to the button's `android:theme` attribute:

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:theme="@style/MyButtonTheme"/>
```

Define the theme in res/values/themes.xml:

```
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

3. Borderless Button: @style/Widget.AppCompat.Button.Borderless

BORDERLESS BUTTON

```
<Button
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_button"/>
```

4. Borderless Colored Button: @style/Widget.AppCompat.Button.Borderless.Colored

BORDERLESS COLORED BUTTON

```
<Button
    style="@style/Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_colored_button"/>
```

Section 37.3: Adding a FloatingActionButton (FAB)

In the material design, a `Floating action button` represents the primary action in an Activity. They are distinguished by a circled icon floating above the UI and have motion behaviors that include morphing,

启动和一个可移动的锚点。

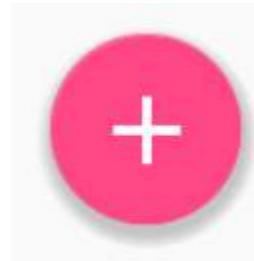
确保在应用的 build.gradle 文件中的 dependencies 下添加以下依赖项：

```
compile 'com.android.support:design:25.3.1'
```

现在将FloatingActionButton添加到你的布局文件中：

```
<android.support.design.widget.FloatingActionButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_margin="16dp"  
    android:src="@drawable/some_icon"/>
```

其中src属性引用了用于浮动操作的图标。



结果应类似如下（假设你的强调色是 Material Pink）：

默认情况下，FloatingActionButton的背景颜色将设置为主题的强调色。另请注意，FloatingActionButton需要周围有边距才能正常工作。推荐的底部边距为手机端16dp，平板端为24dp。

以下是可用于进一步自定义FloatingActionButton的属性（假设在布局顶部声明了命名空间`xmlns:app="http://schemas.android.com/apk/res-auto"`）：

- `app:fabSize`: 可以设置为 `normal` 或 `mini`, 切换为正常大小或较小版本。
- `app:rippleColor`: 设置 FloatingActionButton 的涟漪效果颜色。可以是颜色资源或十六进制字符串。
- `app:elevation`: 可以是字符串、整数、布尔值、颜色值、浮点数、尺寸值。
- `app:useCompatPadding`: 启用兼容内边距。可以是布尔值，如 `true` 或 `false`。设置为 `true` 以便在 API 21 及更高版本上使用兼容内边距，以保持与较旧 API 级别的一致外观。

你可以在这里找到更多关于 FAB 的示例。

第37.4节：RippleDrawable

涟漪触摸效果是在 Android 5.0 (API 级别 21) 中随材质设计引入的，动画由新的 RippleDrawable 类实现。

Drawable 会根据状态变化显示涟漪效果。可以通过调用 `setHotspot(float x, float y)` 并传入对应状态属性标识符，指定涟漪的锚定位置。

版本 ≥ 5.0

一般来说，常规按钮的涟漪效果在 API 21 及以上版本默认启用，对于其他可触摸视图，可以通过指定以下内容来实现：

launching, and a transferring anchor point.

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Now add the FloatingActionButton to your layout file:

```
<android.support.design.widget.FloatingActionButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_margin="16dp"  
    android:src="@drawable/some_icon"/>
```

where the `src` attribute references the icon that should be used for the floating action.



The result should look something like this (presuming your accent color is Material Pink):

By default, the background color of your FloatingActionButton will be set to your theme's accent color. Also, note that a FloatingActionButton requires a margin around it to work properly. The recommended margin for the bottom is 16dp for phones and 24dp for tablets.

Here are properties which you can use to customize the FloatingActionButton further (assuming `xmlns:app="http://schemas.android.com/apk/res-auto` is declared as namespace the top of your layout):

- `app:fabSize`: Can be set to `normal` or `mini` to switch between a normal sized or a smaller version.
- `app:rippleColor`: Sets the color of the ripple effect of your FloatingActionButton. Can be a color resource or hex string.
- `app:elevation`: Can be a string, integer, boolean, color value, floating point, dimension value.
- `app:useCompatPadding`: Enable compat padding. Maybe a boolean value, such as `true` or `false`. Set to `true` to use compat padding on api-21 and later, in order to maintain a consistent look with older api levels.

You can find more examples about FAB here.

Section 37.4: RippleDrawable

Ripple touch effect was introduced with material design in Android 5.0 (API level 21) and the animation is implemented by the new `RippleDrawable` class.

Drawable that shows a ripple effect in response to state changes. The anchoring position of the ripple for a given state may be specified by calling `setHotspot(float x, float y)` with the corresponding state attribute identifier.

Version ≥ 5.0

In general, ripple effect for **regular buttons works by default** in API 21 and above, and for other touchable views, it can be achieved by specifying:

```
android:background="?android:attr/selectableItemBackground">
```

用于视图内的涟漪效果，或者：

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

用于超出视图边界的涟漪效果。

例如，在下图中，

- B1 是一个没有任何背景的按钮，
- B2 设置了 `android:background="?android:attr/selectableItemBackground"`
- B3 设置了 `android:background="?android:attr/selectableItemBackgroundBorderless"`

```
android:background="?android:attr/selectableItemBackground">
```

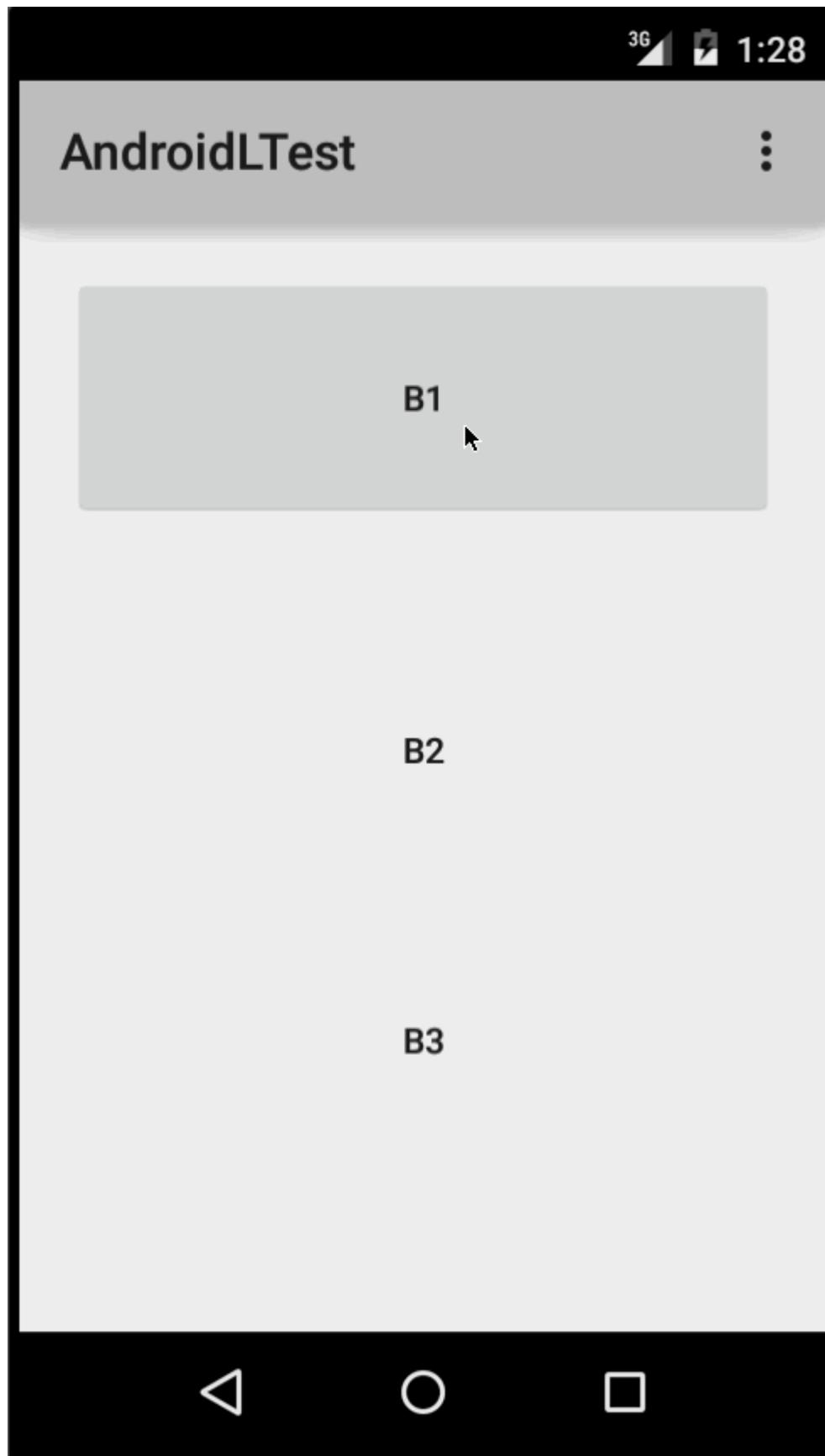
for ripples contained within the view or:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

for ripples that extend beyond the view's bounds.

For example, in the image below,

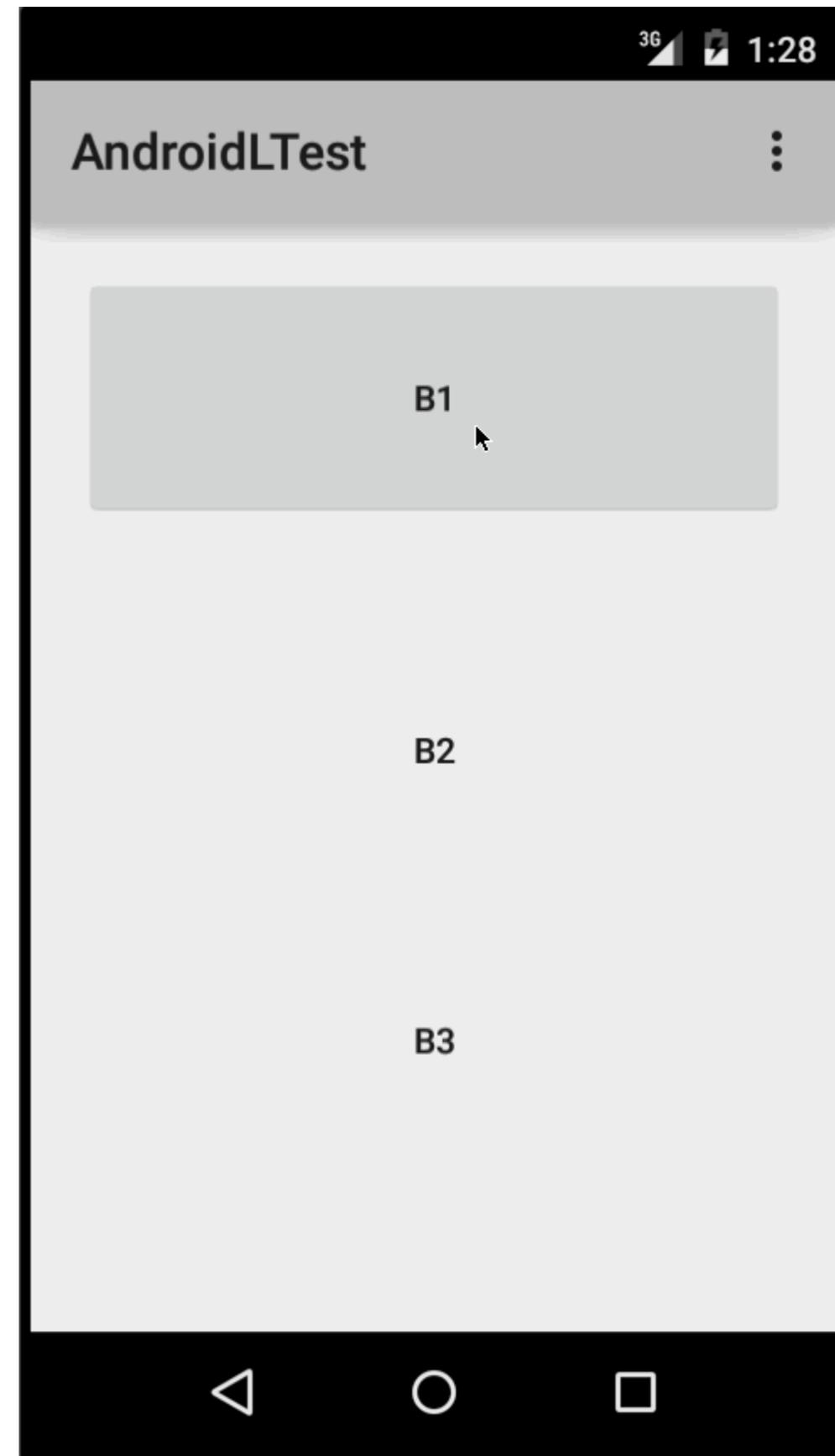
- B1 is a button that does not have any background,
- B2 is set up with `android:background="?android:attr/selectableItemBackground"`
- B3 is set up with `android:background="?android:attr/selectableItemBackgroundBorderless"`



(图片来源：<http://blog.csdn.net/a396901990/article/details/40187203>)

你也可以通过代码实现相同效果：

```
int[] attrs = new int[]{R.attr.selectableItemBackground};  
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);  
int backgroundResource = typedArray.getResourceId(0, 0);
```



(Image courtesy: <http://blog.csdn.net/a396901990/article/details/40187203>)

You can achieve the same in code using:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};  
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);  
int backgroundResource = typedArray.getResourceId(0, 0);
```

```
myView.setBackgroundResource(backgroundResource);
```

涟漪效果也可以通过`android:foreground`属性以同样的方式添加到视图中。如名称所示，如果涟漪添加到前景，涟漪将显示在其添加的任何视图之上（例如`ImageView`，包含多个视图的`LinearLayout`等）。

如果你想自定义视图中的涟漪效果，需要在`drawable`目录下创建一个新的XML文件。

以下是一些示例：

示例1：无边界涟漪

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="#ffff0000" />
```

示例2：带遮罩和背景色的涟漪

```
<ripple android:color="#777777"  
        xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:id="@+id/mask"  
          android:drawable="#ffff00" />  
    <item android:drawable="@android:color/white" />  
</ripple>
```

如果视图view已经指定了背景background，且带有shape、corners及其他标签，要在该视图上添加涟漪效果，请使用mask layer，并将涟漪设置为视图的背景。

示例：

```
<?xml version="1.0" encoding="utf-8"?>  
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="?android:attr/colorControlHighlight">  
    <item android:id="@+id/mask">  
        <shape  
            android:shape="rectangle">  
            solid android:color="#000000"/>  
        <corners  
            android:radius="25dp"/>  
        </shape>  
    </item>  
    <item android:drawable="@drawable/rounded_corners" />  
</ripple>
```

示例3：在可绘制资源上方的涟漪效果

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="#ff0000ff">  
    <item android:drawable="@drawable/my_drawable" />  
</ripple>
```

用法：要将你的ripple xml文件附加到任何视图，按以下方式设置为背景（假设你的ripple文件名为my_ripple.xml）：

```
<View  
    android:id="@+id/myViewId"  
    android:layout_width="wrap_content"
```

```
myView.setBackgroundResource(backgroundResource);
```

Ripples can also be added to a view using the `android:foreground` attribute the same way as above. As the name suggests, in case the ripple is added to the foreground, the ripple will show up above any view it is added to (e.g. `ImageView`, a `LinearLayout` containing multiple views, etc).

If you want to customize the ripple effect into a view, you need to create a new XML file, inside the `drawable` directory.

Here are few examples:

Example 1: An unbounded ripple

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="#ffff0000" />
```

Example 2: Ripple with mask and background color

```
<ripple android:color="#777777"  
        xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:id="@+id/mask"  
          android:drawable="#ffff00" />  
    <item android:drawable="@android:color/white" />  
</ripple>
```

If there is view with a background *already specified* with a shape, corners and any other tags, to add a ripple to that view use a `mask` layer and set the ripple as the background of the view.

Example:

```
<?xml version="1.0" encoding="utf-8"?>  
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="?android:attr/colorControlHighlight">  
    <item android:id="@+id/mask">  
        <shape  
            android:shape="rectangle">  
            solid android:color="#000000"/>  
        <corners  
            android:radius="25dp"/>  
        </shape>  
    </item>  
    <item android:drawable="@drawable/rounded_corners" />  
</ripple>
```

Example 3: Ripple on top a drawable resource

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"  
        android:color="#ff0000ff">  
    <item android:drawable="@drawable/my_drawable" />  
</ripple>
```

Usage: To attach your ripple xml file to any view, set it as background as following (assuming your ripple file is named `my_ripple.xml`):

```
<View  
    android:id="@+id/myViewId"  
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:background="@drawable/my_ripple" />
```

选择器：

如果目标版本是v21或以上，ripple drawable也可以替代颜色状态列表选择器（你也可以将ripple选择器放在drawable-v21文件夹中）：

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="?android:colorControlHighlight">
    <item android:drawable="@drawable/button_normal" />
</ripple>
```

在这种情况下，视图的默认状态颜色将是白色，按下状态将显示波纹drawable。

注意点：使用 ?android:colorControlHighlight 会使波纹颜色与应用中内置波纹的颜色相同。

要仅更改波纹颜色，可以像这样在主题中自定义颜色 android:colorControlHighlight：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
        <item name="android:colorControlHighlight">@color/your_custom_color</item>
    </style>

</resources>
```

然后在活动等中使用此主题。效果如下图所示：

```
        android:layout_height="wrap_content"
        android:background="@drawable/my_ripple" />
```

Selector:

The ripple drawable can also be used in place of color state list selectors if your target version is v21 or above (you can also place the ripple selector in the drawable-v21 folder):

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="?android:colorControlHighlight">
    <item android:drawable="@drawable/button_normal" />
</ripple>
```

In this case, the color of the default state of your view would be white and the pressed state would show the ripple drawable.

Point to note: Using ?android:colorControlHighlight will give the ripple the same color as the built-in ripples in your app.

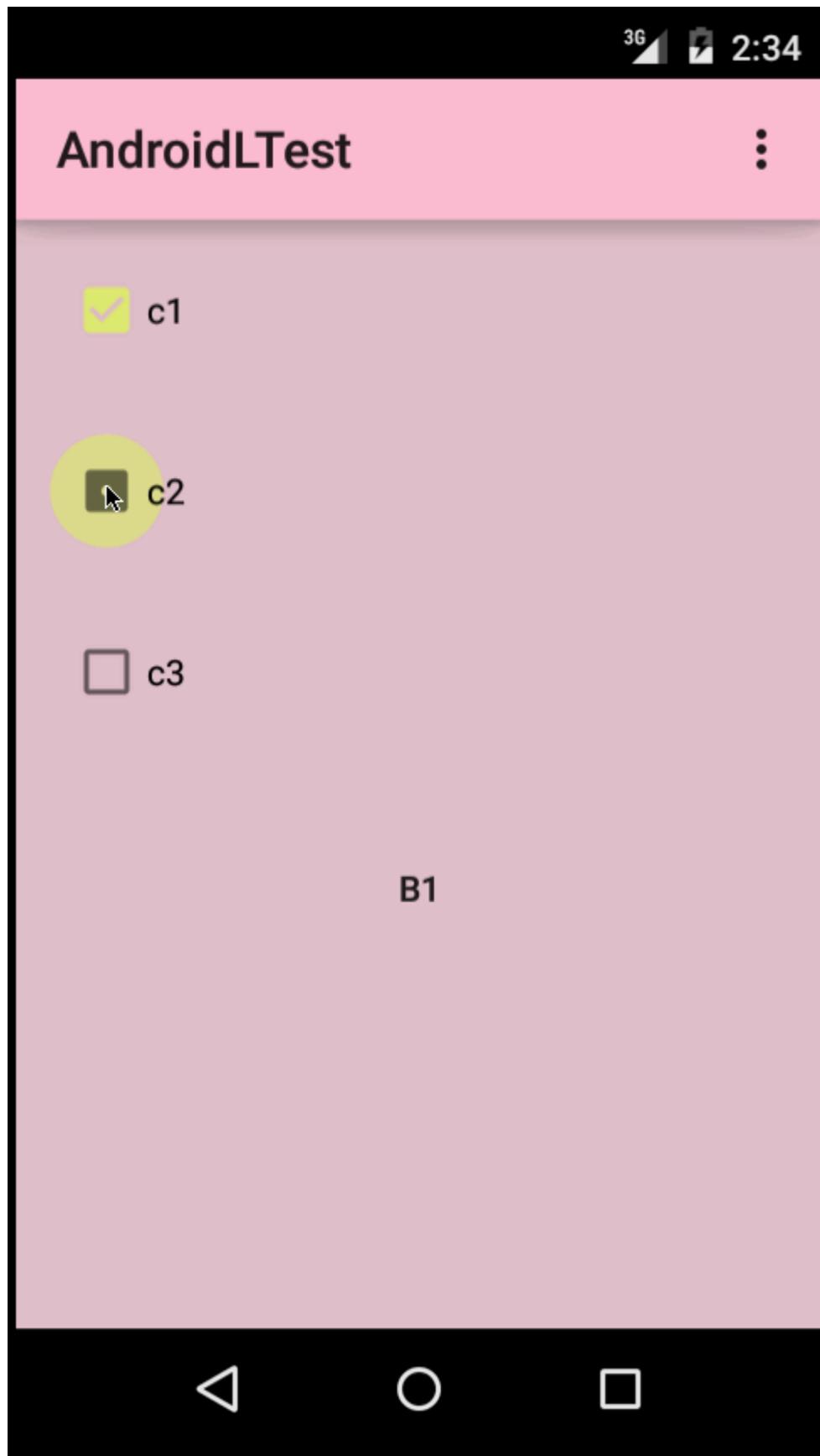
To change just the ripple color, you can customize the color android:colorControlHighlight in your theme like so:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
        <item name="android:colorControlHighlight">@color/your_custom_color</item>
    </style>

</resources>
```

and then use this theme in your activities, etc. The effect would be like the image below:

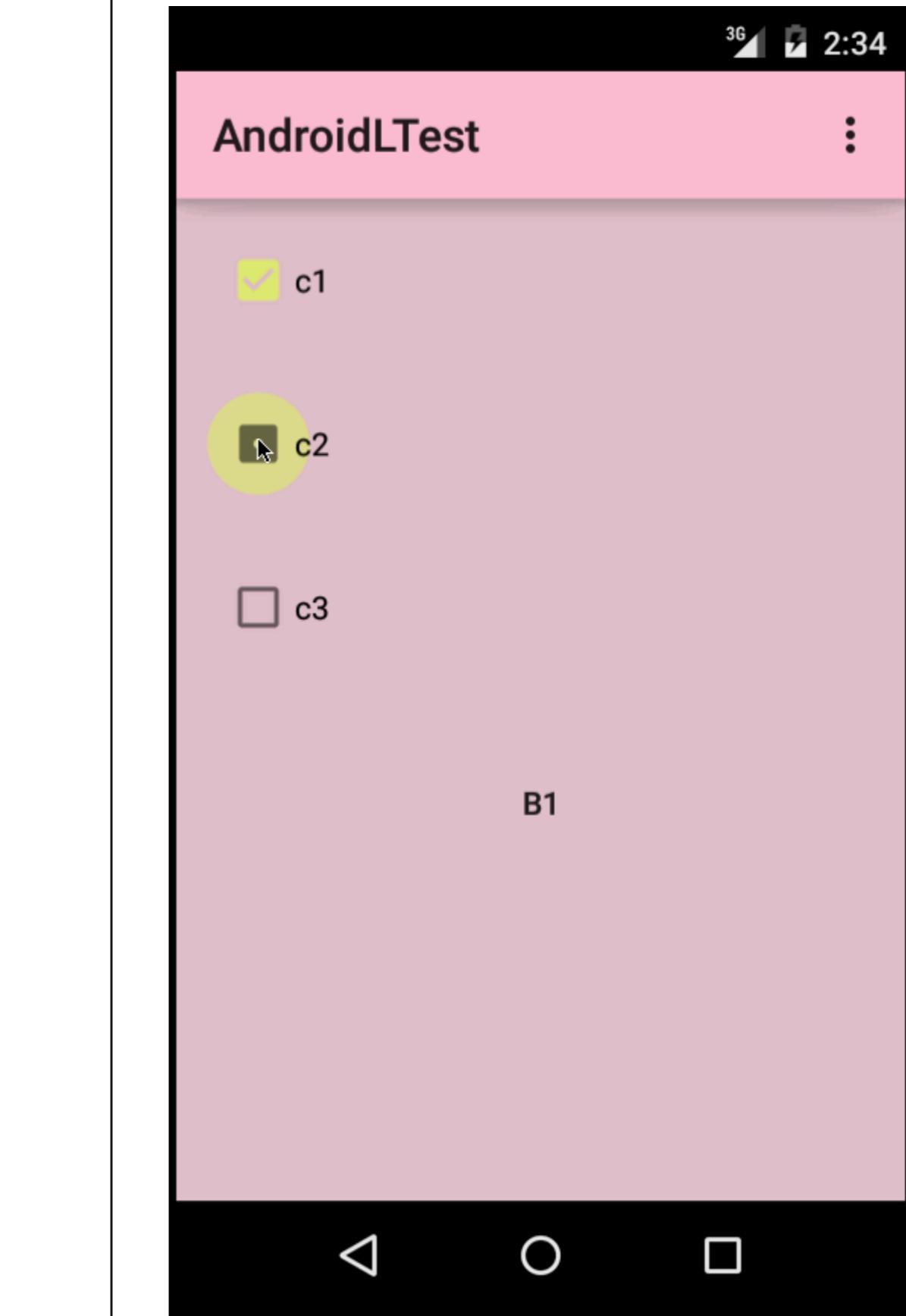


(图片来源：<http://blog.csdn.net/a396901990/article/details/40187203>)

第37.5节：添加TabLayout

TabLayout 提供一个水平布局来显示标签，通常与 ViewPager 一起使用。

确保在应用的build.gradle文件的dependencies部分添加以下依赖：



(Image courtesy: <http://blog.csdn.net/a396901990/article/details/40187203>)

Section 37.5: Adding a TabLayout

TabLayout provides a horizontal layout to display tabs, and is commonly used in conjunction with a ViewPager.

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

现在你可以使用 `TabItem` 类在布局中向 `TabLayout` 添加项目。

例如：

```
<android.support.design.widget.TabLayout  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:id="@+id/tabLayout">  
  
    <android.support.design.widget.TabItem  
        android:text="@string/tab_text_1"  
        android:icon="@drawable/ic_tab_1"/>  
  
    <android.support.design.widget.TabItem  
        android:text="@string/tab_text_2"  
        android:icon="@drawable/ic_tab_2"/>  
  
</android.support.design.widget.TabLayout>
```

添加一个 `OnTabSelectedListener` 以便在 `TabLayout` 中的标签被选中/取消选中/重新选中时收到通知：

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);  
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
    @Override  
    public void onTabSelected(TabLayout.Tab tab) {  
        int position = tab.getPosition();  
        // 切换到该标签对应的视图  
    }  
  
    @Override  
    public void onTabUnselected(TabLayout.Tab tab) {  
  
    }  
  
    @Override  
    public void onTabReselected(TabLayout.Tab tab) {  
  
    }  
});
```

选项卡也可以通过编程方式从 `TabLayout` 中添加或移除。

```
TabLayout.Tab tab = tabLayout.newTab();  
tab.setText(R.string.tab_text_1);  
tab.setIcon(R.drawable.ic_tab_1);  
tabLayout.addTab(tab);  
  
tabLayout.removeTab(tab);  
tabLayout.removeTabAt(0);  
tabLayout.removeAllTabs();
```

`TabLayout` 有两种模式，固定 (fixed) 和可滚动 (scrollable)。

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);  
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

这些也可以在 XML 中应用：

```
compile 'com.android.support:design:25.3.1'
```

Now you can add items to a `TabLayout` in your layout using the `TabItem` class.

For example:

```
<android.support.design.widget.TabLayout  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:id="@+id/tabLayout">  
  
    <android.support.design.widget.TabItem  
        android:text="@string/tab_text_1"  
        android:icon="@drawable/ic_tab_1"/>  
  
    <android.support.design.widget.TabItem  
        android:text="@string/tab_text_2"  
        android:icon="@drawable/ic_tab_2"/>  
  
</android.support.design.widget.TabLayout>
```

Add an `OnTabSelectedListener` to be notified when a tab in the `TabLayout` is selected/unselected/reselected:

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);  
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {  
    @Override  
    public void onTabSelected(TabLayout.Tab tab) {  
        int position = tab.getPosition();  
        // Switch to view for this tab  
    }  
  
    @Override  
    public void onTabUnselected(TabLayout.Tab tab) {  
  
    }  
  
    @Override  
    public void onTabReselected(TabLayout.Tab tab) {  
  
    }  
});
```

Tabs can also be added/removed from the `TabLayout` programmatically.

```
TabLayout.Tab tab = tabLayout.newTab();  
tab.setText(R.string.tab_text_1);  
tab.setIcon(R.drawable.ic_tab_1);  
tabLayout.addTab(tab);  
  
tabLayout.removeTab(tab);  
tabLayout.removeTabAt(0);  
tabLayout.removeAllTabs();
```

`TabLayout` has two modes, fixed and scrollable.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);  
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

These can also be applied in XML:

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tabLayout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:tabMode="fixed|scrollable" />
```

注意：TabLayout 的模式是互斥的，意味着一次只能激活一种模式。

标签指示器颜色是为您的 Material Design 主题定义的强调色。

您可以通过在 styles.xml 中定义自定义样式，然后将该样式应用到您的 TabLayout 来覆盖此颜色：

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">  
    <item name="tabIndicatorColor">@color/your_color</item>  
</style>
```

然后你可以通过以下方式将样式应用到视图：

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tabs"  
    style="@style/MyCustomTabLayoutStyle"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
</android.support.design.widget.TabLayout>
```

第37.6节：设计支持库中的底部面板

底部面板从屏幕底部滑出以显示更多内容。

它们在Android支持库v25.1.0版本中被添加，并支持所有以上版本。

确保在应用的 build.gradle 文件中的 dependencies 下添加以下依赖项：

```
compile 'com.android.support:design:25.3.1'
```

持久性底部面板

你可以通过将BottomSheetBehavior附加到

CoordinatorLayout的子视图来实现持久性底部面板：

```
<android.support.design.widget.CoordinatorLayout >  
  
    <!-- . . . -->  
  
    <LinearLayout  
        android:id="@+id/bottom_sheet"  
        android:elevation="4dp"  
        android:minHeight="120dp"  
        app:behavior_peekHeight="120dp"  
        ...  
        app:layout_behavior="android.support.design.widget.BottomSheetBehavior">  
  
    <!-- . . . -->  
  
    </LinearLayout>  
  
</android.support.design.widget.CoordinatorLayout>
```

然后在你的代码中可以使用以下方式创建引用：

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tabLayout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:tabMode="fixed|scrollable" />
```

Note: the TabLayout modes are mutually exclusive, meaning only one can be active at a time.

The tab indicator color is the accent color defined for your Material Design theme.

You can override this color by defining a custom style in styles.xml and then applying the style to your TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">  
    <item name="tabIndicatorColor">@color/your_color</item>  
</style>
```

Then you can apply the style to the view using:

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tabs"  
    style="@style/MyCustomTabLayoutStyle"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
</android.support.design.widget.TabLayout>
```

Section 37.6: Bottom Sheets in Design Support Library

Bottom sheets slide up from the bottom of the screen to reveal more content.

They were added to the Android Support Library in v25.1.0 version and supports above all the versions.

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Persistent Bottom Sheets

You can achieve a Persistent Bottom Sheet attaching a BottomSheetBehavior to a child View of a CoordinatorLayout:

```
<android.support.design.widget.CoordinatorLayout >  
  
    <!-- . . . -->  
  
    <LinearLayout  
        android:id="@+id/bottom_sheet"  
        android:elevation="4dp"  
        android:minHeight="120dp"  
        app:behavior_peekHeight="120dp"  
        ...  
        app:layout_behavior="android.support.design.widget.BottomSheetBehavior">  
  
    <!-- . . . -->  
  
    </LinearLayout>  
  
</android.support.design.widget.CoordinatorLayout>
```

Then in your code you can create a reference using:

```
// 带有 BottomSheetBehavior 的视图
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

你可以使用 `setState()` 方法设置 `BottomSheetBehavior` 的状态：

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

你可以使用以下状态之一：

- `STATE_COLLAPSED`：此折叠状态为默认状态，仅显示底部布局的一部分。
高度可以通过 `app:behavior_peekHeight` 属性控制（默认值为 0）
- `STATE_EXPANDED`：底部弹出层的完全展开状态，此时整个底部弹出层可见
(如果其高度小于包含的 CoordinatorLayout) 或整个 CoordinatorLayout 被填满
- `STATE_HIDDEN`：默认禁用（可通过 `app:behavior_hideable` 属性启用），启用后
允许用户向下滑动底部弹出层以完全隐藏底部弹出层

此外，要通过点击您选择的视图（比如一个按钮）来打开或关闭 BottomSheet，以下是切换
弹出层行为并更新视图的方法。

```
mButton = (Button) findViewById(R.id.button_2);
// 点击按钮时，我们监控弹出层的状态
mButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_EXPANDED) {
            // 如果已展开，则折叠它（设置为 Peek 模式）。
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_COLLAPSED);
            mButton.setText(R.string.button2_hide);
        } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_COLLAPSED) {
            // 如果折叠，则完全隐藏它。
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_HIDDEN);
            mButton.setText(R.string.button2);
        } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_HIDDEN) {
            // 如果隐藏，则根据需要折叠或展开。
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
            mButton.setText(R.string.button2_peek);
        }
    }
});
```

但是 BottomSheet 行为还有一个功能，用户可以通过拖动手势向上或向下滑动它。在这种情况下，如果面板状态发生
变化，我们可能无法更新依赖的视图（如上面的按钮）。为此，你可能希望接收状态变化的回调，因此可以添加一
个

`BottomSheetCallback` 来监听用户的滑动事件：

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
        // 响应状态变化并通知视图当前状态
    }
    @Override
    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        // 响应拖动事件并对依赖视图进行动画或透明度处理
    }
})
```

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

You can set the state of your `BottomSheetBehavior` using the `setState()` method:

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

You can use one of these states:

- `STATE_COLLAPSED`: this collapsed state is the default and shows just a portion of the layout along the bottom.
The height can be controlled with the `app:behavior_peekHeight` attribute (defaults to 0)
- `STATE_EXPANDED`: the fully expanded state of the bottom sheet, where either the whole bottom sheet is visible
(if its height is less than the containing CoordinatorLayout) or the entire CoordinatorLayout is filled
- `STATE_HIDDEN`: disabled by default (and enabled with the `app:behavior_hideable` attribute), enabling this
allows users to swipe down on the bottom sheet to completely hide the bottom sheet

Further to open or close the BottomSheet on click of a View of your choice, A Button let's say, here is how to toggle
the sheet behavior and update view.

```
mButton = (Button) findViewById(R.id.button_2);
// On Button click we monitor the state of the sheet
mButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_EXPANDED) {
            // If expanded then collapse it (setting in Peek mode).
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_COLLAPSED);
            mButton.setText(R.string.button2_hide);
        } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_COLLAPSED) {
            // If Collapsed then hide it completely.
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_HIDDEN);
            mButton.setText(R.string.button2);
        } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_HIDDEN) {
            // If hidden then Collapse or Expand, as the need be.
            mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
            mButton.setText(R.string.button2_peek);
        }
    }
});
```

But BottomSheet behavior also has a feature where user can interact with the swipe UP or Down it with a DRAG motion. In such a case, we might not be able to update the dependent View (like the button above) If the Sheet state has changed. For that matter, you'd like to receive callbacks of state changes, hence you can add a `BottomSheetCallback` to listen to user swipe events:

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
        // React to state change and notify views of the current state
    }
    @Override
    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        // React to dragging events and animate views or transparency of dependent views
    }
})
```

```
});
```

如果您只希望底部弹出层在折叠（COLLAPSED）和展开（EXPANDED）模式下可见，并且永远不隐藏（HIDE），请使用：

```
mBottomSheetBehavior2.setHideable(false);
```

底部弹出层对话框片段

您也可以在底部弹出层中显示一个BottomSheetDialogFragment来替代视图。为此，您首先需要创建一个继承自BottomSheetDialogFragment的新类。

在setupDialog()方法中，您可以加载一个新的布局文件，并获取Activity中容器视图的BottomSheetBehavior。一旦获得该行为，您可以创建并关联一个BottomSheetCallback，以便在弹出层隐藏时关闭该Fragment。

```
public class BottomSheetDialogFragmentExample extends BottomSheetDialogFragment {  
  
    private BottomSheetBehavior.BottomSheetCallback mBottomSheetBehaviorCallback = new  
    BottomSheetBehavior.BottomSheetCallback() {  
  
        @Override  
        public void onStateChanged(@NonNull View bottomSheet, int newState) {  
            if (newState == BottomSheetBehavior.STATE_HIDDEN) {  
                dismiss();  
            }  
        }  
  
        @Override  
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {}  
    };  
  
    @Override  
    public void setupDialog(Dialog dialog, int style) {  
        super.setupDialog(dialog, style);  
        View contentView = View.inflate(getContext(), R.layout.fragment_bottom_sheet, null);  
        dialog.setContentView(contentView);  
  
        CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) ((View)  
        contentView.getParent()).getLayoutParams();  
        CoordinatorLayout.Behavior behavior = params.getBehavior();  
  
        if( behavior != null && behavior instanceof BottomSheetBehavior ) {  
            ((BottomSheetBehavior) behavior).setBottomSheetCallback(mBottomSheetBehaviorCallback);  
        }  
    }  
}
```

最后，您可以调用 Fragment 的实例的 show() 方法，将其显示在底部弹出窗中。

```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();  
bottomSheetDialogFragment.show(getSupportFragmentManager(), bottomSheetDialogFragment.getTag());
```

您可以在专门的主题中找到更多详细信息

```
});
```

And if you only want your Bottom Sheet to be visible only in COLLAPSED and EXPANDED mode toggles and never HIDE use:

```
mBottomSheetBehavior2.setHideable(false);
```

Bottom Sheet DialogFragment

You can also display a [BottomSheetDialogFragment](#) in place of a View in the bottom sheet. To do this, you first need to create a new class that extends BottomSheetDialogFragment.

Within the setupDialog() method, you can inflate a new layout file and retrieve the BottomSheetBehavior of the container view in your Activity. Once you have the behavior, you can create and associate a [BottomSheetCallback](#) with it to dismiss the Fragment when the sheet is hidden.

```
public class BottomSheetDialogFragmentExample extends BottomSheetDialogFragment {  
  
    private BottomSheetBehavior.BottomSheetCallback mBottomSheetBehaviorCallback = new  
    BottomSheetBehavior.BottomSheetCallback() {  
  
        @Override  
        public void onStateChanged(@NonNull View bottomSheet, int newState) {  
            if (newState == BottomSheetBehavior.STATE_HIDDEN) {  
                dismiss();  
            }  
        }  
  
        @Override  
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {}  
    };  
  
    @Override  
    public void setupDialog(Dialog dialog, int style) {  
        super.setupDialog(dialog, style);  
        View contentView = View.inflate(getContext(), R.layout.fragment_bottom_sheet, null);  
        dialog.setContentView(contentView);  
  
        CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) ((View)  
        contentView.getParent()).getLayoutParams();  
        CoordinatorLayout.Behavior behavior = params.getBehavior();  
  
        if( behavior != null && behavior instanceof BottomSheetBehavior ) {  
            ((BottomSheetBehavior) behavior).setBottomSheetCallback(mBottomSheetBehaviorCallback);  
        }  
    }  
}
```

Finally, you can call show() on an instance of your Fragment to display it in the bottom sheet.

```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();  
bottomSheetDialogFragment.show(getSupportFragmentManager(), bottomSheetDialogFragment.getTag());
```

You can find more details in the dedicated topic

第37.7节：应用 AppCompat 主题

AppCompat 支持库提供了用于构建符合Material Design 规范的应用主题。Activity 继承AppCompatActivity时，也需要使用父主题为Theme.AppCompat的主题。

第一步是自定义主题的颜色调色板，以自动为应用着色。

在应用的res/styles.xml中，你可以定义：

```
<!-- 继承自 AppCompat 主题 -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- 应用栏的应用品牌色 -->
    <item name="colorPrimary">#2196f3</item>

    <!-- 状态栏和上下文应用栏的深色变体 -->
    <item name="colorPrimaryDark">#1976d2</item>

    <!-- 主题的 UI 控件，如复选框和文本框 -->
    <item name="colorAccent">#f44336</item>
</style>
```

除了背景较暗的Theme.AppCompat，你还可以使用Theme.AppCompat.Light或

Theme.AppCompat.Light.DarkActionBar。

您可以使用自己的颜色自定义主题。不错的选择可以参考Material设计规范颜色图表，以及Material调色板。“500”色是作为主色的好选择（此例中为蓝色500）；选择同一色调的“700”作为深色；并选择不同色调的一个色作为强调色。主色用于应用的工具栏及其在概览（最近应用）屏幕中的显示，深色变体用于给状态栏着色，强调色用于突出显示某些控件。

创建此主题后，将其应用到应用程序的AndroidManifest.xml中，并且也应用到任何特定的活动中。这对于应用AppTheme.NoActionBar主题非常有用，该主题允许你实现非默认的工具栏配置。

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
```

你也可以使用android:theme和ThemeOverlay主题将主题应用到单个视图。例如，对于Toolbar：

```
<android.support.v7.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />
```

或者一个Button：

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:theme="@style/MyButtonTheme" />
```

Section 37.7: Apply an AppCompat theme

The AppCompat support library provides themes to build apps with the [Material Design specification](#). A theme with a parent of Theme.AppCompat is also required for an Activity to extend AppCompatActivity.

The first step is to customize your theme's [color palette](#) to automatically colorize your app. In your app's res/styles.xml you can define:

```
<!-- inherit from the AppCompat theme -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">#2196f3</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">#1976d2</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">#f44336</item>
</style>
```

Instead of Theme.AppCompat, which has a dark background, you can also use Theme.AppCompat.Light or Theme.AppCompat.Light.DarkActionBar.

You can customize the theme with your own colours. Good choices are in the [Material design specification colour chart](#), and [Material Palette](#). The "500" colours are good choices for primary (blue 500 in this example); choose "700" of the same hue for the dark one; and an a shade from a different hue as the accent colour. The primary colour is used for your app's toolbar and its entry in the overview (recent apps) screen, the darker variant to tint the status bar, and the accent colour to highlight some controls.

After creating this theme, apply it to your app in the AndroidManifest.xml and also apply the theme to any particular activity. This is useful for applying a AppTheme.NoActionBar theme, which lets you implement non-default toolbar configurations.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

You can also apply themes to individual Views using android:theme and a ThemeOverlay theme. For example with a Toolbar:

```
<android.support.v7.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />
```

or a Button:

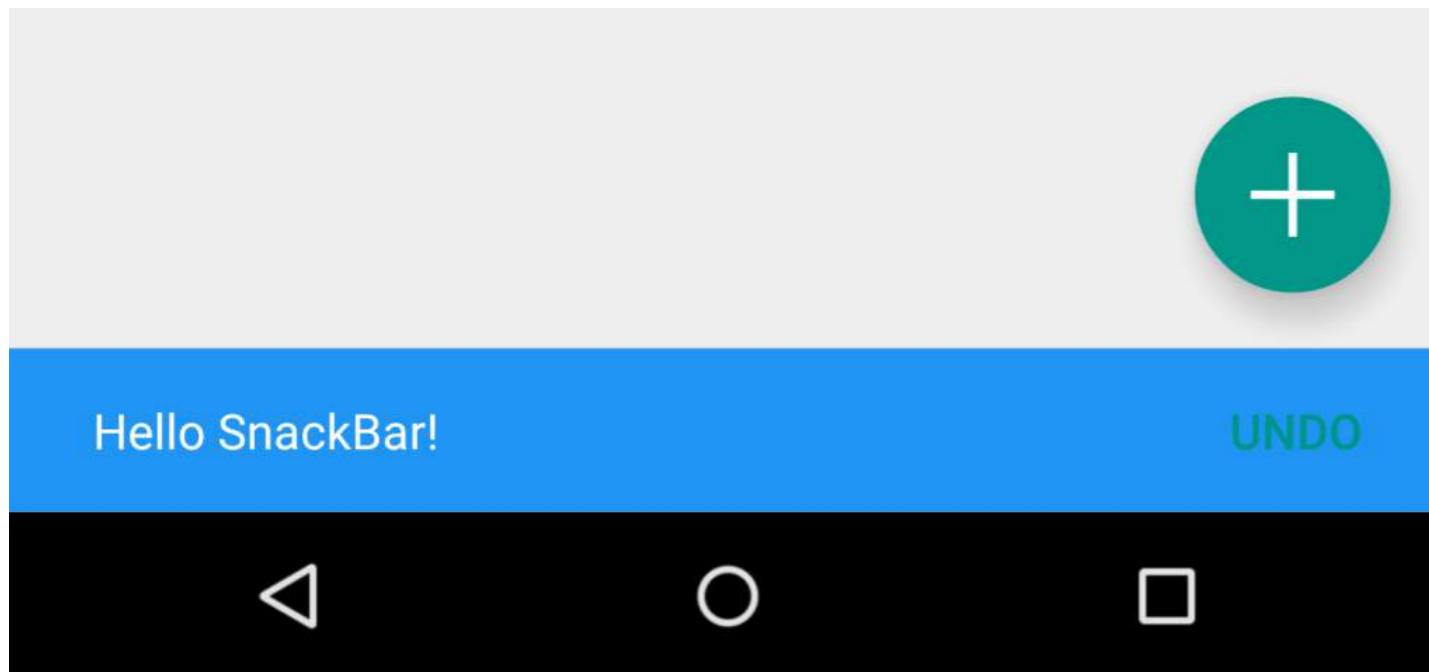
```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:theme="@style/MyButtonTheme" />
```

```
<!-- res/values/themes.xml -->
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

第37.8节：添加Snackbar

Material Design 的主要特性之一是新增了一个Snackbar，理论上它取代了之前的 Toast。根据 Android 文档：

Snackbar 包含与执行操作直接相关的单行文本。它们可以包含一个文本操作，但不包含图标。Toast 主要用于系统消息。它们也显示在屏幕底部，但不能被滑动离开屏幕。



在 Android 中仍然可以使用 Toast 向用户显示消息，然而如果你决定在应用中采用 Material Design，建议你实际使用 Snackbar。Snackbar 不是作为屏幕上的覆盖层显示，而是从底部弹出。

实现方法如下：

```
Snackbar snackbar = Snackbar
    .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

关于显示 Snackbar 的时长，我们有类似于 Toast 的选项，或者可以设置自定义的毫秒数：

- LENGTH_SHORT
- LENGTH_LONG
- 长度不定
- setDuration() (自版本22.2.1起)

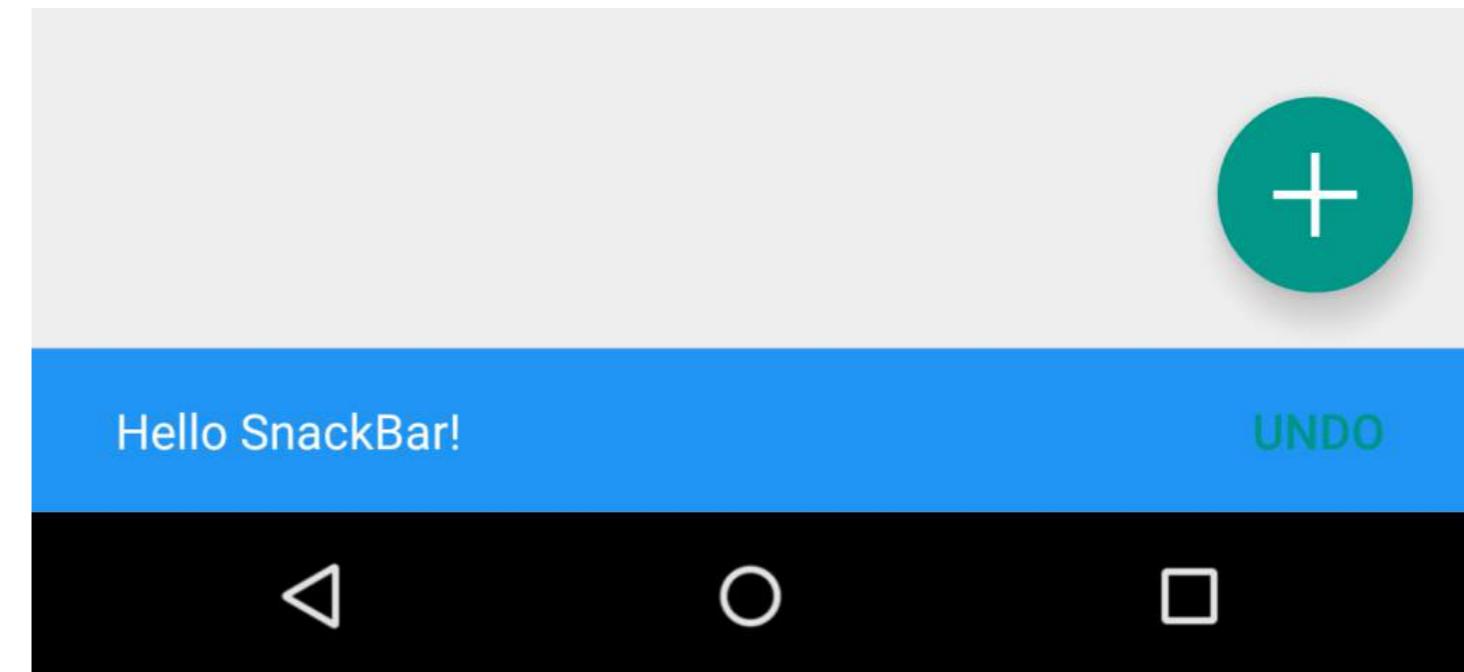
你还可以为你的Snackbar添加动态功能，比如ActionCallback或自定义颜色。但在自定义Snackbar时，请务必注意Android提供的设计指南。

```
<!-- res/values/themes.xml -->
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

Section 37.8: Add a Snackbar

One of the main features in Material Design is the addition of a Snackbar, which in theory replaces the previous Toast. As per the Android documentation:

Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons. Toasts are primarily used for system messaging. They also display at the bottom of the screen, but may not be swiped off-screen.



Toasts can still be used in Android to display messages to users, however if you have decided to opt for material design usage in your app, it is recommended that you actually use a snackbar. Instead of being displayed as an overlay on your screen, a Snackbar pops from the bottom.

Here is how it is done:

```
Snackbar snackbar = Snackbar
    .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

As for the length of time to show the Snackbar, we have the options similar to the ones offered by a Toast or we could set a custom duration in milliseconds:

- LENGTH_SHORT
- LENGTH_LONG
- LENGTH_INDEFINITE
- setDuration() (since version 22.2.1)

You can also add dynamic features to your Snackbar such as ActionCallback or custom color. However do pay attention to the [design guideline](#) offered by Android when customising a Snackbar.

实现Snackbar有一个限制。你要实现的视图的父布局必须是一个CoordinatorLayout。这样才能实现从底部弹出的实际效果。

这是在布局xml文件中定义CoordinatorLayout的方法：

```
<android.support.design.widget.CoordinatorLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/coordinatorLayout"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity">  
  
    //布局中的其他控件放在这里。  
  
</android.support.design.widget.CoordinatorLayout>
```

然后需要在你的Activity的onCreate方法中定义CoordinatorLayout，之后在创建Snackbar时使用它。

有关Snackbar的更多信息，请查看官方文档或文档中的专门章节。

第37.9节：添加导航抽屉

导航抽屉用于在应用中导航到顶级目的地。

确保你已经在build.gradle文件的dependencies部分添加了设计支持库：

```
dependencies {  
    // ...  
    compile 'com.android.support:design:25.3.1'  
}
```

接下来，在你的XML布局资源文件中添加DrawerLayout和NavigationView。

DrawerLayout只是一个华丽的容器，允许实际的导航抽屉NavigationView从屏幕的左侧或右侧滑出。

注意：对于移动设备，标准抽屉尺寸为320dp。

```
<!-- res/layout/activity_main.xml -->  
<android.support.v4.widget.DrawerLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:app="http://schemas.android.com/apk/res-auto"  
        xmlns:tools="http://schemas.android.com/tools"  
        android:id="@+id/navigation_drawer_layout"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:fitsSystemWindows="true"  
        tools:openDrawer="start">  
    <! -- 你可以使用"end"从右侧打开抽屉 -->  
  
<android.support.design.widget.CoordinatorLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:fitsSystemWindows="true">  
  
    <android.support.design.widget.AppBarLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:theme="@style/AppTheme.AppBarOverlay">
```

Implementing the Snackbar has one limitation however. The parent layout of the view you are going to implement a Snackbar in needs to be a CoordinatorLayout. This is so that the actual popup from the bottom can be made.

This is how to define a CoordinatorLayout in your layout xml file:

```
<android.support.design.widget.CoordinatorLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/coordinatorLayout"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity">  
  
    //any other widgets in your layout go here.  
  
</android.support.design.widget.CoordinatorLayout>
```

The CoordinatorLayout then needs to be defined in your Activity's onCreate method, and then used when creating the Snackbar itself.

For more information about about the Snackbar, please check the [official documentation](#) or the dedicated topic in the documentation.

Section 37.9: Add a Navigation Drawer

[Navigation Drawers](#) are used to navigate to top-level destinations in an app.

Make sure that you have added design support library in your build.gradle file under dependencies:

```
dependencies {  
    // ...  
    compile 'com.android.support:design:25.3.1'  
}
```

Next, add the DrawerLayout and NavigationView in your XML layout resource file.

The DrawerLayout is just a fancy container that allows the NavigationView, the actual navigation drawer, to slide out from the left or right of the screen. Note: for mobile devices, the standard drawer size is 320dp.

```
<!-- res/layout/activity_main.xml -->  
<android.support.v4.widget.DrawerLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:app="http://schemas.android.com/apk/res-auto"  
        xmlns:tools="http://schemas.android.com/tools"  
        android:id="@+id/navigation_drawer_layout"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:fitsSystemWindows="true"  
        tools:openDrawer="start">  
    <! -- You can use "end" to open drawer from the right side -->  
  
<android.support.design.widget.CoordinatorLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:fitsSystemWindows="true">  
  
    <android.support.design.widget.AppBarLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:theme="@style/AppTheme.AppBarOverlay">
```

```

<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    app:popupTheme="@style/AppTheme.PopupOverlay" />

</android.support.design.widget.AppBarLayout>

</android.support.design.widget.CoordinatorLayout>

<android.support.design.widget.NavigationView
    android:id="@+id/navigation_drawer"
    android:layout_width="320dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/drawer_header"
    app:menu="@menu/navigation_menu" />

</android.support.v4.widget.DrawerLayout>

```

现在，如果你愿意，可以创建一个header文件，作为导航抽屉的顶部。这用于使抽屉看起来更加优雅。

```

<!-- res/layout/drawer_header.xml -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="190dp">

    <ImageView
        android:id="@+id/header_image"
        android:layout_width="140dp"
        android:layout_height="120dp"
        android:layout_centerInParent="true"
        android:scaleType="centerCrop"
        android:src="@drawable/image" />

    <TextView
        android:id="@+id/header_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/header_image"
        android:text="用户名"
        android:textSize="20sp" />

</RelativeLayout>

```

它在NavigationView标签中的app:headerLayout="@layout/drawer_header"属性中被引用。

该app:headerLayout会自动将指定的布局填充到头部。也可以在

运行时通过以下方式实现：

```

// 查找导航视图
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
// 在运行时填充头部视图
View headerLayout = navigationView.inflateHeaderView(R.layout.drawer_header);

```

要自动填充符合材质设计规范的导航抽屉导航项，请创建一个菜单

```

<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    app:popupTheme="@style/AppTheme.PopupOverlay" />

</android.support.design.widget.AppBarLayout>

</android.support.design.widget.CoordinatorLayout>

<android.support.design.widget.NavigationView
    android:id="@+id/navigation_drawer"
    android:layout_width="320dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/drawer_header"
    app:menu="@menu/navigation_menu" />

</android.support.v4.widget.DrawerLayout>

```

Now, if you wish, create a **header file** that will serve as the top of your navigation drawer. This is used to give a much more elegant look to the drawer.

```

<!-- res/layout/drawer_header.xml -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="190dp">

    <ImageView
        android:id="@+id/header_image"
        android:layout_width="140dp"
        android:layout_height="120dp"
        android:layout_centerInParent="true"
        android:scaleType="centerCrop"
        android:src="@drawable/image" />

    <TextView
        android:id="@+id/header_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/header_image"
        android:text="User name"
        android:textSize="20sp" />

</RelativeLayout>

```

It is referenced in the NavigationView tag in the app:headerLayout="@layout/drawer_header" attribute. This app:headerLayout inflates the specified layout into the header automatically. This can alternatively be done at runtime with:

```

// Lookup navigation view
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
// Inflate the header view at runtime
View headerLayout = navigationView.inflateHeaderView(R.layout.drawer_header);

```

To automatically populate your navigation drawer with material design-compliant navigation items, create a menu

根据需要添加文件和项目。注意：虽然项目的图标不是必需的，但在[Material Design 规范中建议使用](#)。

它在app中的NavigationView标签中被引用：menu="@menu/navigation_menu"属性。

```
<!-- res/menu/menu_drawer.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_item_1"
        android:title="项目 #1"
        android:icon="@drawable/ic_nav_1" />
    <item
        android:id="@+id/nav_item_2"
        android:title="项目 #2"
        android:icon="@drawable/ic_nav_2" />
    <item
        android:id="@+id/nav_item_3"
        android:title="项目 #3"
        android:icon="@drawable/ic_nav_3" />
    <item
        android:id="@+id/nav_item_4"
        android:title="项目 #4"
        android:icon="@drawable/ic_nav_4" />
</menu>
```

要将项目分组，请将它们放入带有android:title属性的另一个<item>中嵌套的<menu>，或用<group>标签包裹它们。

现在布局完成，接下来编写Activity代码：

```
// 查找导航视图
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // 获取项目ID以确定用户点击后的操作
        int itemId = item.getItemId();
        // 使用新的Intent响应导航抽屉的选择
startActivity(new Intent(this, OtherActivity.class));
        return true;
    }
});

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.navigation_drawer_layout);
// 打开和关闭时自动动画导航抽屉所必需
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, "Open navigation drawer",
"Close navigation drawer");
// 这两个字符串不会显示给用户，但请确保将它们放入单独的strings.xml文件中。

drawer.addDrawerListener(toggle);
toggle.syncState();
```

您现在可以在NavigationView的头部视图中随意操作

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerView.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerView.findViewById(R.id.header_image);
// 设置导航头部文本
headerTextView.setText("用户名");
// 设置导航头部图片
```

file and add items as needed. Note: while icons for items aren't required, they are suggested in the [Material Design specification](#).

It is referenced in the NavigationView tag in the app:menu="@menu/navigation_menu" attribute.

```
<!-- res/menu/menu_drawer.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_item_1"
        android:title="Item #1"
        android:icon="@drawable/ic_nav_1" />
    <item
        android:id="@+id/nav_item_2"
        android:title="Item #2"
        android:icon="@drawable/ic_nav_2" />
    <item
        android:id="@+id/nav_item_3"
        android:title="Item #3"
        android:icon="@drawable/ic_nav_3" />
    <item
        android:id="@+id/nav_item_4"
        android:title="Item #4"
        android:icon="@drawable/ic_nav_4" />
</menu>
```

To separate items into groups, put them into a <menu> nested in another <item> with an android:title attribute or wrap them with the <group> tag.

Now that the layout is done, move on to the Activity code:

```
// Find the navigation view
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Get item ID to determine what to do on user click
        int itemId = item.getItemId();
        // Respond to Navigation Drawer selections with a new Intent
startActivity(new Intent(this, OtherActivity.class));
        return true;
    }
});

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.navigation_drawer_layout);
// Necessary for automatically animated navigation drawer upon open and close
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, "Open navigation drawer",
"Close navigation drawer");
// The two Strings are not displayed to the user, but be sure to put them into a separate
strings.xml file.
drawer.addDrawerListener(toggle);
toggle.syncState();
```

You can now do whatever you want in the header view of the NavigationView

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerView.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerView.findViewById(R.id.header_image);
// Set navigation header text
headerTextView.setText("User name");
// Set navigation header image
```

```
headerImageView.setImageResource(R.drawable.header_image);
```

头部视图的行为就像其他任何View一样，因此一旦您使用findViewById()并在布局文件中添加其他View，就可以设置其中任何元素的属性。

您可以在专门的主题中找到更多细节和示例。

第37.10节：如何使用TextInputLayout

确保在应用的build.gradle文件的dependencies部分添加以下依赖：

```
compile 'com.android.support:design:25.3.1'
```

当输入值时，将EditText中的提示作为浮动标签显示。

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/form_username" />

</android.support.design.widget.TextInputLayout>
```

要显示带有TextInputLayout的密码显示眼睛图标，可以使用以下代码：

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/input_layout_current_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/current_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/current_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

其中app:passwordToggleEnabled="true" 和 android:inputType="textPassword" 参数是必需的。

app 应使用命名空间 `xmlns:app="http://schemas.android.com/apk/res-auto"`

您可以在专门的主题中找到更多详细信息和示例。

```
headerImageView.setImageResource(R.drawable.header_image);
```

The header view behaves like any other View, so once you use `findViewById()` and add some other Views to your layout file, you can set the properties of anything in it.

You can find more details and examples in the dedicated topic.

Section 37.10: How to use TextInputLayout

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Show the hint from an EditText as a floating label when a value is entered.

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/form_username" />

</android.support.design.widget.TextInputLayout>
```

For displaying the password display eye icon with TextInputLayout, we can make use of the following code:

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/input_layout_current_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/current_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/current_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

where `app:passwordToggleEnabled="true"` & `android:inputType="textPassword"` parameters are required.

app should use the namespace `xmlns:app="http://schemas.android.com/apk/res-auto"`

You can find more details and examples in the dedicated topic.

第38章：资源

第38.1节：定义颜色

颜色通常存储在名为colors.xml的资源文件中，位于/res/values/文件夹内。

它们由<color>元素定义：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="blackOverlay">#66000000</color>
</resources>
```

颜色由每个颜色通道的十六进制颜色值表示(0 - FF)，格式如下之一：

- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB

图例

- A - 透明通道 - 0值表示完全透明，FF值表示不透明
- R - 红色通道
- G - 绿色通道
- B - 蓝色通道

定义的颜色可以在XML中使用以下语法@color/颜色名称

例如：

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

在代码中使用颜色

这些示例假设this是一个Activity引用。也可以使用Context引用代替。

版本 ≥ 1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

版本 < 6.0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

在上述声明中，colorPrimary、colorPrimaryDark和colorAccent用于定义Material设计颜色，这些颜色将在styles.xml中定义自定义Android主题时使用。它们在使用Android Studio创建新项目时会自动添加。

Chapter 38: Resources

Section 38.1: Define colors

Colors are usually stored in a resource file named `colors.xml` in the `/res/values/` folder.

They are defined by `<color>` elements:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="blackOverlay">#66000000</color>
</resources>
```

Colors are represented by hexadecimal color values for each color channel (0 - FF) in one of the formats:

- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB

Legend

- A - alpha channel - 0 value is fully transparent, FF value is opaque
- R - red channel
- G - green channel
- B - blue channel

Defined colors can be used in XML with following syntax `@color/name_of_the_color`

For example:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Using colors in code

These examples assume `this` is an Activity reference. A Context reference can be used in its place as well.

Version ≥ 1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

Version < 6.0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

In above declaration `colorPrimary`, `colorPrimaryDark` and `colorAccent` are used to define Material design colors that will be used in defining custom Android theme in `styles.xml`. They are automatically added when new project is created with Android Studio.

第38.2节：颜色透明度 (Alpha) 级别

十六进制不透明度值

透明度(%)	十六进制值
100%	FF
95%	F2
90%	E6
85%	D9
80%	CC
75%	BF
70%	B3
65%	A6
60%	99
55%	8C
50%	80
45%	73
40%	66
35%	59
30%	4D
25%	40
20%	33
15%	26
10%	1A
5%	0D
0%	00

如果你想将45%设置为红色。

```
<color name="red_with_alpha_45">#73FF0000</color>
```

红色的十六进制值 - #FF0000

你可以在前缀中添加73表示45%的不透明度 - #73FF0000

第38.3节：定义字符串复数形式

为了区分复数和单数字符串，你可以在你的strings.xml文件中定义复数，并列出不同的数量，如下面的示例所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="hello_people">
        <item quantity="one">向%d个人问好</item>
        <item quantity="other">向%d个人问好</item>
    </plurals>
</resources>
```

可以通过使用Resources类的getQuantityString()方法从Java代码中访问此定义，示例如下：

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

这里，第一个参数R.plurals.hello_people是资源名称。第二个参数（本例中为3）

Section 38.2: Color Transparency(Alpha) Level

Hex Opacity Values

Alpha(%)	Hex Value
100%	FF
95%	F2
90%	E6
85%	D9
80%	CC
75%	BF
70%	B3
65%	A6
60%	99
55%	8C
50%	80
45%	73
40%	66
35%	59
30%	4D
25%	40
20%	33
15%	26
10%	1A
5%	0D
0%	00

If you want to set 45% to red color.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

hex value for red - #FF0000

You can add 73 for 45% opacity in prefix - #73FF0000

Section 38.3: Define String Plurals

To differentiate between plural and singular strings, you can define a plural in your *strings.xml* file and list the different quantities, as shown in the example below:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="hello_people">
        <item quantity="one">Hello to %d person</item>
        <item quantity="other">Hello to %d people</item>
    </plurals>
</resources>
```

This definition can be accessed from Java code by using the `getQuantityString()` method of the `Resources` class, as shown in the following example:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Here, the first parameter `R.plurals.hello_people` is the resource name. The second parameter (3 in this example)

用于选择正确的quantity字符串。第三个参数（本例中同样为3）是格式参数，将用于替换格式说明符%d。

可能的quantity值（按字母顺序列出）有：

```
few  
many  
one  
other  
two  
zero
```

需要注意的是，并非所有语言环境都支持每种quantity的表示。例如，中文没有“one”项的概念。英语没有“zero”项，因为它在语法上与“other”相同。IDE会将不支持的quantity实例标记为Lint警告，但如果使用它们不会导致编译错误。

第38.4节：定义字符串

字符串通常存储在资源文件strings.xml中。它们使用<string> XML元素定义。

strings.xml的目的是支持国际化。你可以为每个语言ISO代码定义一个strings.xml文件。因此，当系统查找字符串“app_name”时，会先检查对应当前语言的xml文件，如果未找到，则查找默认的strings.xml文件。这意味着你可以选择只本地化部分字符串，而不必全部本地化。

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">你好，世界应用</string>  
    <string name="hello_world">你好，世界！</string>  
</resources>
```

一旦在 XML 资源文件中定义了字符串，应用程序的其他部分就可以使用它。

应用程序的 XML 项目文件可以通过引用 @string/string_name 来使用 <string> 元素。例如，应用程序的 [manifest](#) (/manifests/AndroidManifest.xml) 文件在 Android Studio 中默认包含以下行：

```
android:label="@string/app_name"
```

这告诉 android 查找名为 "app_name" 的 <string> 资源，作为应用程序安装或在启动器中显示时的名称。

另一个在 android 中使用 XML 文件中的 <string> 资源的场景是在布局文件中。例如，以下表示一个 TextView，显示我们之前定义的 hello_world 字符串：

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello_world"/>
```

你也可以从应用程序的 java 部分访问 <string> 资源。要在 Activity 类中调用上面相同的 hello_world 字符串，使用：

is used to pick the correct quantity string. The third parameter (also 3 in this example) is the format argument that will be used for substituting the format specifier %d.

Possible quantity values (listed in alphabetical order) are:

```
few  
many  
one  
other  
two  
zero
```

It is important to note that not all locales support every denomination of quantity. For example, the Chinese language does not have a concept of one item. English does not have a zero item, as it is grammatically the same as other. Unsupported instances of quantity will be flagged by the IDE as Lint warnings, but won't cause compilation errors if they are used.

Section 38.4: Define strings

Strings are typically stored in the resource file strings.xml. They are defined using a <string> XML element.

The purpose of strings.xml is to allow internationalisation. You can define a strings.xml for each language iso code. Thus when the system looks for the string 'app_name' it first checks the xml file corresponding to the current language, and if it is not found, looks for the entry in the default strings.xml file. This means you can choose to only localise some of your strings while not others.

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">Hello World App</string>  
    <string name="hello_world">Hello World!</string>  
</resources>
```

Once a string is defined in an XML resource file, it can be used by other parts of the app.

An app's XML project files can use a <string> element by referring to @string/string_name. For example, an app's [manifest](#) (/manifests/AndroidManifest.xml) file includes the following line by default in Android Studio:

```
android:label="@string/app_name"
```

This tells android to look for a <string> resource called "app_name" to use as the name for the app when it is installed or displayed in a launcher.

Another time you would use a <string> resource from an XML file in android would be in a layout file. For example, the following represents a TextView which displays the hello_world string we defined earlier:

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello_world"/>
```

You can also access <string> resources from the java portion of your app. To recall our same hello_world string from above within an Activity class, use:

```
String helloWorld = getString(R.string.hello_world);
```

第 38.5 节：定义尺寸

尺寸通常存储在名为 dimens.xml 的资源文件中。它们使用 `<dimen>` 元素定义。

res/values/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

您可以使用不同的单位：

- **sp** : 与缩放无关的像素。用于字体。
- **dp** : 与密度无关的像素。用于其他所有内容。
- **pt** : 点
- **px** : 像素
- **mm** : 毫米
- **in** : 英寸

现在可以使用语法 `@dimen/name_of_the_dimension` 在 XML 中引用尺寸。

例如：

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/large_padding">
</RelativeLayout>
```

第38.6节：strings.xml中的字符串格式化

在strings.xml文件中定义字符串也允许字符串格式化。唯一的注意事项是，字符串需要像下面代码中那样处理，而不是简单地附加到布局中。

```
<string name="welcome_trainer">你好，宝可梦训练师 %1$s！你已经捕捉了 %2$d 只宝可梦。</string>
String welcomePokemonTrainerText = getString(R.string.welcome_trainer, tranerName, pokemonCount);
```

在上述示例中，

%1\$s

'%' 与普通字符分开，

'1' 表示第一个参数，

'\$' 用作参数编号和类型之间的分隔符，

's' 表示字符串类型 ('d' 用于整数)

```
String helloWorld = getString(R.string.hello_world);
```

Section 38.5: Define dimensions

Dimensions are typically stored in a resource file names `dimens.xml`. They are defined using a `<dimen>` element.

res/values/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

You can use different units :

- **sp** : Scale-independent Pixels. For fonts.
- **dp** : Density-independent Pixels. For everything else.
- **pt** : Points
- **px** : Pixels
- **mm** : Millimeters
- **in** : Inches

Dimensions can now be referenced in XML with the syntax `@dimen/name_of_the_dimension`.

For example:

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/large_padding">
</RelativeLayout>
```

Section 38.6: String formatting in strings.xml

Defining Strings in the strings.xml file also allows for string formatting. The only caveat is that the String will need to be dealt with in code like below, versus simply attaching it to a layout.

```
<string name="welcome_trainer">Hello Pokémon Trainer, %1$s! You have caught %2$d Pokémon.</string>
String welcomePokemonTrainerText = getString(R.string.welcome_trainer, tranerName, pokemonCount);
```

In above example,

%1\$s

'%' separates from normal characters,

'1' denotes first parameter,

'\$' is used as separator between parameter number and type,

's' denotes string type ('d' is used for integer)

注意 `getString()` 是 `Context` 或 `Resources` 的方法，即你可以直接在 `Activity` 实例中使用它，否则你可以分别使用 `getActivity().getString()` 或 `getContext().getString()`。

第38.7节：定义整数数组

为了定义一个整数数组，请在资源文件

`res/values/filename.xml` 中编写

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

例如

`res/values/arrays.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="fibo">
        <item>@integer/zero</item>
        <item>@integer/one</item>
        <item>@integer/one</item>
        <item>@integer/two</item>
        <item>@integer/three</item>
        <item>@integer/five</item>
    </integer-array>
</resources>
```

并且在 Java 中这样使用

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG", Arrays.toString(values));
```

输出

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

第38.8节：定义颜色状态列表

颜色状态列表可以用作颜色，但会根据所使用视图的状态而变化。

要定义颜色状态列表，请在 `res/color/foo.xml` 中创建资源文件

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#888888" android:state_enabled="false"/>
    <item android:color="@color/lightGray" android:state_selected="false"/>
    <item android:color="@android:color/white" />
</selector>
```

项目按照定义的顺序进行评估，第一个其指定状态与视图的当前状态匹配的项目将被使用。因此，最好在最后指定一个通配项，不带任何状态选择器。

每个项目可以使用颜色字面值，或者引用其他地方定义的颜色。

Note that `getString()` is a method of `Context` or `Resources`, i.e. you can use it directly within an `Activity` instance, or else you may use `getActivity().getString()` or `getContext().getString()` respectively.

Section 38.7: Define integer array

In order to define an integer array write in a resources file

`res/values/filename.xml`

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

for example

`res/values/arrays.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="fibo">
        <item>@integer/zero</item>
        <item>@integer/one</item>
        <item>@integer/one</item>
        <item>@integer/two</item>
        <item>@integer/three</item>
        <item>@integer/five</item>
    </integer-array>
</resources>
```

and use it from java like

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG", Arrays.toString(values));
```

Output

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Section 38.8: Define a color state list

Color state lists can be used as colors, but will change depending on the state of the view they are used for.

To define one, create a resource file in `res/color/foo.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#888888" android:state_enabled="false"/>
    <item android:color="@color/lightGray" android:state_selected="false"/>
    <item android:color="@android:color/white" />
</selector>
```

Items are evaluated in the order they are defined, and the first item whose specified states match the current state of the view is used. So it's a good practice to specify a catch-all at the end, without any state selectors specified.

Each item can either use a color literal, or reference a color defined somewhere else.

第38.9节：9补丁

9补丁是可拉伸的图像，其中可拉伸的区域由透明边框上的黑色标记定义。

这里有一个很棒的教程。[_____](#)

尽管年代久远，它仍然非常有价值，帮助我们许多人深入理解了9补丁机制。

不幸的是，最近那个页面曾一度下线（现在又重新上线了）。

因此，我们需要在可靠的服务器上为安卓开发者保存该页面的实体副本。

就在这里。

安卓UI的9补丁简单指南 2011年5月18日

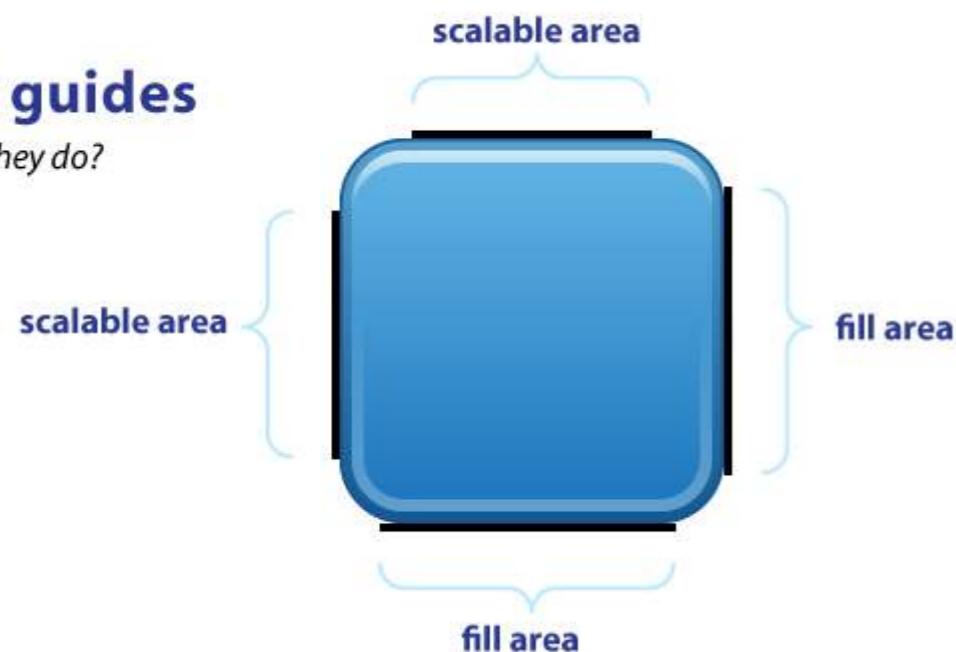
在我开发第一个安卓应用时，发现9补丁（又称9.png）既令人困惑又文档不足。

过了一会儿，我终于弄明白了它是如何工作的，决定做点东西帮助别人理解它。

基本上，9-patch 使用 PNG 透明度来实现一种高级的 9 切片 (9-slice) 或 scale9。引导线是画在图像边缘的直线、1 像素宽的黑线，用来定义图像的缩放和填充。通过将图像命名为 file name.9.png，Android 会识别 9.png 格式，并使用黑色引导线来缩放和填充你的位图。

这里是一个基本的引导图：

9-patch guides what do they do?



如你所见，图像的每一边都有引导线。顶部和左侧的引导线用于缩放图像（即 9 切片），而右侧和底部的引导线定义填充区域。

黑色引导线会从图像中裁剪/移除-它们不会在应用中显示。引导线必须只有一像素宽，所以如果你想要一个 48x48 的按钮，你的 PNG 实际上应该是 50x50。任何超过一像素宽度的线条都会保留在图像中。（我的示例中为了更好看清，引导线宽度是 4 像素，实际上应该只有 1 像素）。

你的引导线必须是纯黑色 (#000000)。即使颜色有一点点差异 (#000001) 或透明度不同，也会导致失败

Section 38.9: 9 Patches

9 Patches are **stretchable** images in which the areas which can be stretched are defined by black markers on a transparent border.

There is a great tutorial [here](#).

Despite being so old, it's still so valuable and it helped many of us to deeply understand the 9 patch gear.

Unfortunately, recently that page has been put down for a while (it's currently up again).

Hence, the need to have a physical copy of that page for android developers on our reliable server/s.

Here it is.

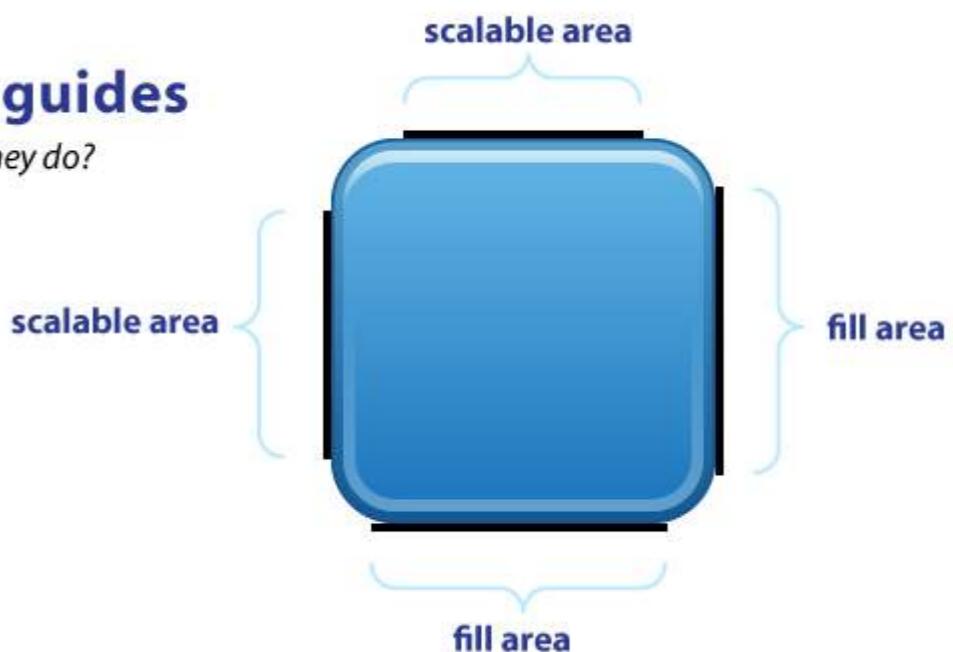
A SIMPLE GUIDE TO 9-PATCH FOR ANDROID UI May 18, 2011

While I was working on my first Android app, I found 9-patch (aka 9.png) to be confusing and poorly documented. After a little while, I finally picked up on how it works and decided to throw together something to help others figure it out.

Basically, 9-patch uses png transparency to do an advanced form of 9-slice or scale9. The guides are straight, 1-pixel black lines drawn on the edge of your image that define the scaling and fill of your image. By naming your image file name.9.png, Android will recognize the 9.png format and use the black guides to scale and fill your bitmaps.

Here's a basic guide map:

9-patch guides what do they do?



As you can see, you have guides on each side of your image. The TOP and LEFT guides are for scaling your image (i.e. 9-slice), while the RIGHT and BOTTOM guides define the fill area.

The black guide lines are cut-off/removed from your image – they won't show in the app. Guides must only be one pixel wide, so if you want a 48x48 button, your png will actually be 50x50. Anything thicker than one pixel will remain part of your image. (My examples have 4-pixel wide guides for better visibility. They should really be only 1-pixel).

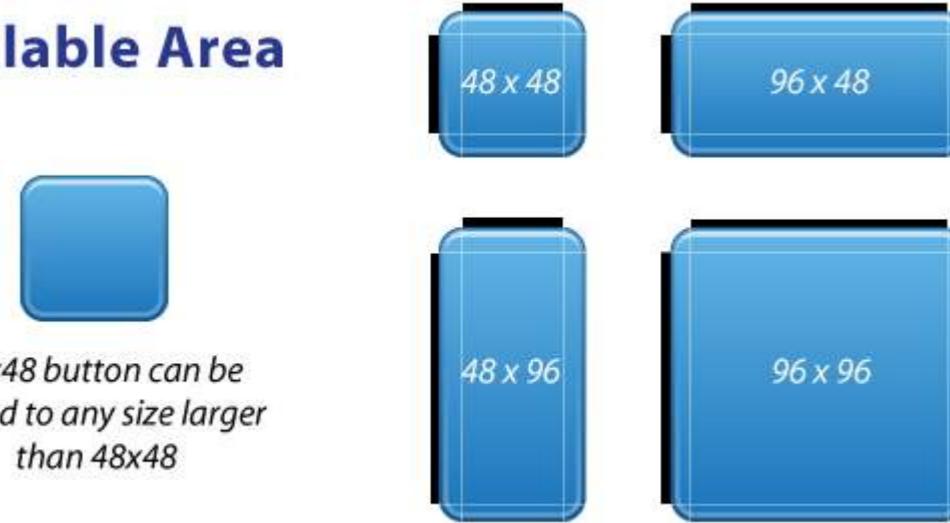
Your guides must be solid black (#000000). Even a slight difference in color (#000001) or alpha will cause it to fail

并且正常拉伸失败。这种失败不会明显表现出来*，它是无声失败！是的，真的。现在你知道了。

另外你还要记住，单像素轮廓的剩余区域必须完全透明。这包括图像的四个角-这些角必须始终是透明的。这可能比你想象的更容易出问题。例如，如果你在 Photoshop 中缩放图像，它会添加抗锯齿像素，其中可能包含几乎不可见的像素，这也会导致失败*。如果必须在 Photoshop 中缩放，请在“图像大小”弹出菜单底部的“重采样图像”下拉菜单中使用“最近邻”设置，以保持引导线的锐利边缘。

* (2012 年 1 月更新) 这实际上是最新开发工具包中的一个“修复”。之前它会表现为所有其他图像和资源突然出错，而不是实际损坏的 9-patch 图像。

Scalable Area



顶部和左侧导线用于定义图像中可缩放的部分-左侧用于缩放高度，顶部用于缩放宽度。以按钮图像为例，这意味着按钮可以在黑色部分内水平和垂直拉伸，而其他部分，如角落，将保持原始大小。这使您能够拥有可以缩放到任意大小并保持统一外观的按钮。

需要注意的是，9-patch 图片不会缩小-它们只会放大。所以最好从尽可能小的尺寸开始。

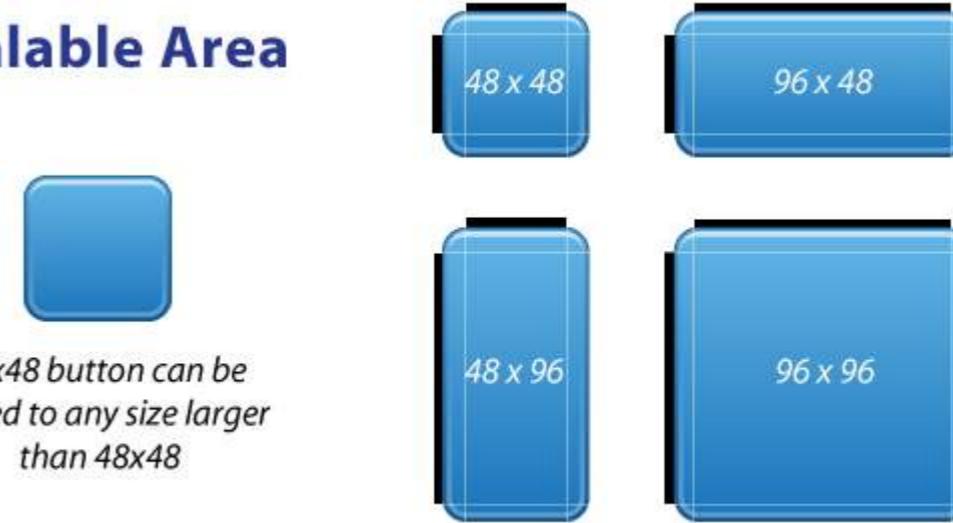
此外，你可以在缩放线的中间部分留空。例如，如果你的按钮中间有一个锋利的光泽边缘，你可以在左侧导线的中间留出几像素。图像的水平中心轴不会缩放，只有其上下部分会缩放，因此你的锋利光泽边缘不会出现抗锯齿或模糊。

and stretch normally. This failure won't be obvious either*, it fails silently! Yes. Really. Now you know.

Also you should keep in mind that remaining area of the one-pixel outline must be completely transparent. This includes the four corners of the image – those should always be clear. This can be a bigger problem than you realize. For example, if you scale an image in Photoshop it will add anti-aliased pixels which may include almost-invisible pixels which will also cause it to fail*. If you must scale in Photoshop, use the Nearest Neighbor setting in the Resample Image pulldown menu (at the bottom of the Image Size pop-up menu) to keep sharp edges on your guides.

*(updated 1/2012) This is actually a “fix” in the latest dev kit. Previously it would manifest itself as all of your other images and resources suddenly breaking, not the actually broken 9-patch image.

Scalable Area

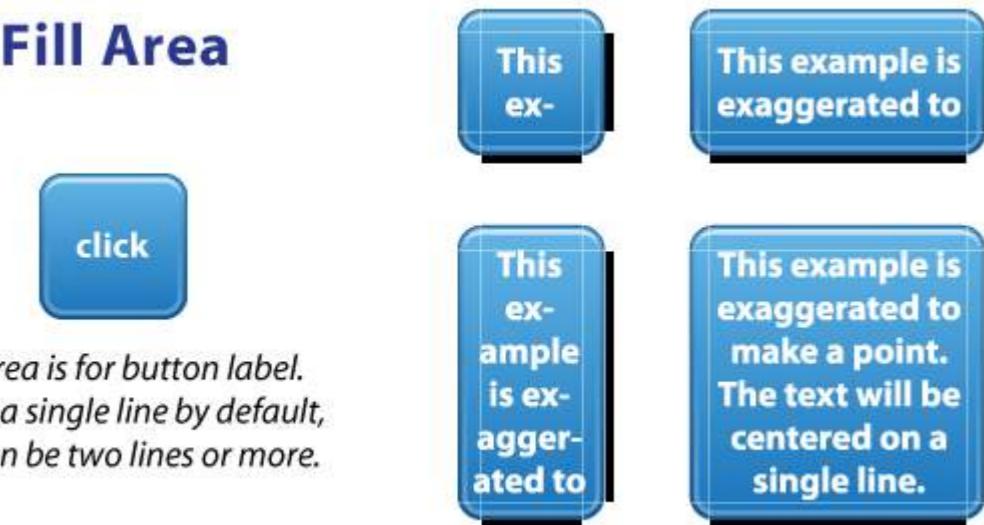


The TOP and LEFT guides are used to define the scalable portion of your image – LEFT for scaling height, TOP for scaling width. Using a button image as an example, this means the button can stretch horizontally and vertically within the black portion and everything else, such as the corners, will remain the same size. This allows you to have buttons that can scale to any size and maintain a uniform look.

It's important to note that 9-patch images don't scale down – they only scale up. So it's best to start as small as possible.

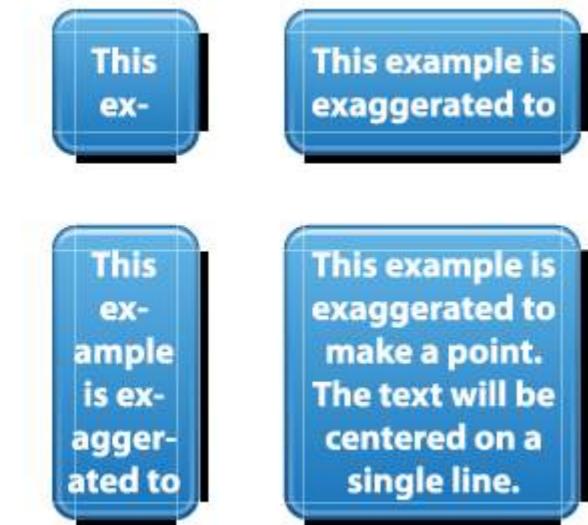
Also, you can leave out portions in the middle of the scale line. So for example, if you have a button with a sharp glossy edge across the middle, you can leave out a few pixels in the middle of the LEFT guide. The center horizontal axis of your image won't scale, just the parts above and below it, so your sharp gloss won't get anti-aliased or fuzzy.

Fill Area



Fill area is for button label.
Text is a single line by default,
but can be two lines or more.

Fill Area



Fill area is for button label.
Text is a single line by default,
but can be two lines or more.

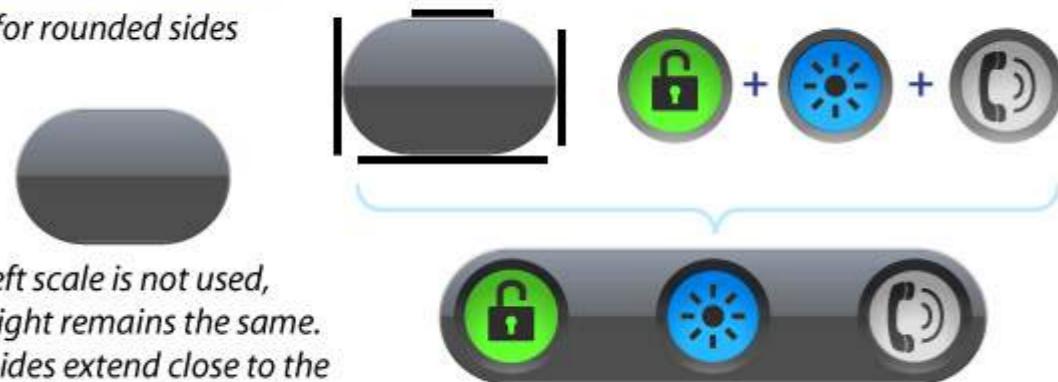
填充区域导线是可选的，用于定义文本标签等内容的区域。填充决定了图像内放置文本、图标或其他元素的空间大小。
9-patch 不仅适用于按钮，也适用于背景图像。

上面的按钮和标签示例是夸张的，仅用于解释填充的概念-标签并不完全准确。说实话，我还没有体验过 Android 如何处理多行标签，因为按钮标签通常是一行文本。

最后，这里有一个很好的示例，展示了缩放和填充导线如何变化，比如带有背景图像和完全圆角边的 LinearLayout

Scale & Fill

for rounded sides



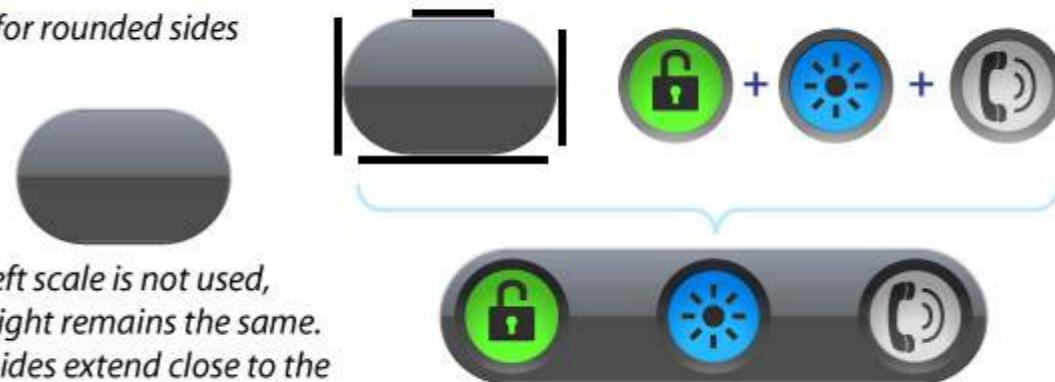
Left scale is not used,
so height remains the same.
Fill guides extend close to the
ends to make room for fitted icons.

在这个示例中，左侧导线没有被使用，但我们仍然需要有导线。背景图像不会垂直缩放；它只会水平缩放（基于顶部导线）。看填充导线，右侧和底部导线延伸到了图像圆角边缘之外。这让我可以将圆形按钮放置得靠近背景边缘，形成紧凑贴合的效果。

就是这样。9-patch 一旦掌握起来非常简单。它不是完美的缩放方式，但填充区域和多行缩放导线比传统的 9-slice 和 scale9 提供了更多灵活性。试试看，你很快就会明白的。

Scale & Fill

for rounded sides



Left scale is not used,
so height remains the same.
Fill guides extend close to the
ends to make room for fitted icons.

With this example, the LEFT guide isn't used but we're still required to have a guide. The background image don't scale vertically; it just scales horizontally (based on the TOP guide). Looking at the fill guides, the RIGHT and BOTTOM guides extend beyond where they meet the image's curved edges. This allows me to place my round buttons close to the edges of the background for a tight, fitted look.

So that's it. 9-patch is super easy, once you get it. It's not a perfect way to do scaling, but the fill-area and multi-line scale-guides does offer more flexibility than traditional 9-slice and scale9. Give it a try and you'll figure it out quickly.

第38.10节：获取资源时避免“已弃用”警告

使用 Android API 23 或更高版本时，经常会遇到这样的情况：

```
context.getResources().getColor(R.color.colorPrimaryDark);  
init();  
'getColor(int)' is deprecated more... (Ctrl+F1)
```

这种情况是由Android API在获取资源方面的结构变化引起的。

现在应该使用以下函数：

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

但android.support.v4库有另一种解决方案。

在build.gradle文件中添加以下依赖：

```
com.android.support:support-v4:24.0.0
```

然后支持库中的所有方法都可用：

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);  
ContextCompat.getDrawable(context, R.drawable.btn_check);  
ContextCompat.getColorStateList(context, R.color.colorPrimary);  
DrawableCompat.setTint(drawable);  
ContextCompat.getColor(context, R.color.colorPrimaryDark);
```

此外，还可以使用支持库中的更多方法：

```
ViewCompat.setElevation(textView, 1F);  
ViewCompat.animate(textView);  
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);  
...
```

第38.11节：使用strings.xml文件

字符串资源为您的应用程序提供文本字符串，并可选择性地进行文本样式和格式设置。有三种类型的资源可以为您的应用程序提供字符串：

字符串

提供单个字符串的XML资源。

语法：

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="string_name">text_string</string>  
</resources>
```

在布局中使用此字符串：

```
<TextView  
    android:layout_width="fill_parent"
```

Section 38.10: Getting resources without "deprecated" warnings

Using the Android API 23 or higher, very often such situation can be seen:

```
context.getResources().getColor(R.color.colorPrimaryDark);  
init();  
'getColor(int)' is deprecated more... (Ctrl+F1)
```

This situation is caused by the structural change of the Android API regarding getting the resources.

Now the function:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

should be used. But the android.support.v4 library has another solution.

Add the following dependency to the build.gradle file:

```
com.android.support:support-v4:24.0.0
```

Then all methods from support library are available:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);  
ContextCompat.getDrawable(context, R.drawable.btn_check);  
ContextCompat.getColorStateList(context, R.color.colorPrimary);  
DrawableCompat.setTint(drawable);  
ContextCompat.getColor(context, R.color.colorPrimaryDark);
```

Moreover more methods from support library can be used:

```
ViewCompat.setElevation(textView, 1F);  
ViewCompat.animate(textView);  
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);  
...
```

Section 38.11: Working with strings.xml file

A string resource provides text strings for your application with optional text styling and formatting. There are three types of resources that can provide your application with strings:

String

XML resource that provides a single string.

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="string_name">text_string</string>  
</resources>
```

And to use this string in layout:

```
<TextView  
    android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="@string/string_name" />
```

字符串数组

提供字符串数组的XML资源。

语法：

```
<resources>
<string-array name="planets_array">
    <item>水星</item>
    <item>金星</item>
    <item>地球</item>
    <item>火星</item>
</string-array>
```

用法

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

数量字符串（复数形式）

携带不同复数形式字符串的XML资源。

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
        name="plural_name">
        <item
            quantity=["zero" | "one" | "two" | "few" | "many" | "other"]
            >text_string</item>
    </plurals>
</resources>
```

用法：

```
int count = getNumberofsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

第38.12节：定义字符串数组

为了定义字符串数组，请在资源文件

`res/values/filename.xml`中编写

```
<string-array name="string_array_name">
    <item>text_string</item></string-array>
    <item>@string/string_id</item>
</string-array>
```

例如

```
        android:layout_height="wrap_content"
        android:text="@string/string_name" />
```

String Array

XML resource that provides an array of strings.

Syntax:

```
<resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

Usage

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Quantity Strings (Plurals)

XML resource that carries different strings for pluralization.

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
        name="plural_name">
        <item
            quantity=["zero" | "one" | "two" | "few" | "many" | "other"]
            >text_string</item>
    </plurals>
</resources>
```

Usage:

```
int count = getNumberofsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Section 38.12: Define string array

In order to define a string array write in a resources file

`res/values/filename.xml`

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

for example

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_example">
        <item>@string/app_name</item>
        <item>@string/hello_world</item>
    </string-array>
</resources>
```

并且在 Java 中这样使用

```
String[] strings = getResources().getStringArray(R.array.string_array_example);
Log.i("TAG", Arrays.toString(strings));
```

输出

```
I/TAG: [HelloWorld, Hello World!]
```

第38.13节：定义整数

整数通常存储在名为integers.xml的资源文件中，但文件名可以任意选择。

每个整数通过使用<integer>元素定义，如以下文件所示：

res/values/integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

现在可以使用语法@integer/name_of_the_integer在XML中引用整数，如以下示例所示：

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max" />
```

第38.14节：定义菜单资源并在Activity/Fragment中使用

在res/menu中定义菜单

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/first_item_id"
        android:orderInCategory="100"
        android:title="@string/first_item_string"
        android:icon="@drawable/first_item_icon"
        app:showAsAction="ifRoom" />
```

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_example">
        <item>@string/app_name</item>
        <item>@string/hello_world</item>
    </string-array>
</resources>
```

and use it from java like

```
String[] strings = getResources().getStringArray(R.array.string_array_example);
Log.i("TAG", Arrays.toString(strings));
```

Output

```
I/TAG: [HelloWorld, Hello World!]
```

Section 38.13: Define integers

Integers are typically stored in a resource file named `integers.xml`, but the file name can be chosen arbitrarily. Each integer is defined by using an `<integer>` element, as shown in the following file:

res/values/integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Integers can now be referenced in XML with the syntax `@integer/name_of_the_integer`, as shown in the following example:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max" />
```

Section 38.14: Define a menu resource and use it inside Activity/Fragment

Define a menu in `res/menu`

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/first_item_id"
        android:orderInCategory="100"
        android:title="@string/first_item_string"
        android:icon="@drawable/first_item_icon"
        app:showAsAction="ifRoom" />
```

```
<item  
    android:id="@+id/second_item_id"  
    android:orderInCategory="110"  
    android:title="@string/second_item_string"  
    android:icon="@drawable/second_item_icon"  
    app:showAsAction="ifRoom"/>  
  
</menu>
```

有关更多配置选项, 请参阅 : [菜单资源](#)

在**Activity**中 :

```
@Override  
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
    //重写以定义菜单资源  
    inflater.inflate(R.menu.menu_resource_id, menu);  
    super.onCreateOptionsMenu(menu, inflater);  
}  
  
@Override  
public void onPrepareOptionsMenu(Menu menu) {  
    //重写以准备菜单项 (设置可见性、更改文本、更改图标...)  
    super.onPrepareOptionsMenu(menu);  
}  
  
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    //重写以处理菜单项  
    int menuItemId = item.getItemId();  
    switch (menuItemId) {  
        case: R.id.first_item_id  
            return true; //如果已处理则返回 true  
    }  
    return super.onOptionsItemSelected(item);  
}
```

在显示视图时调用上述方法, 调用 `getActivity().invalidateOptionsMenu()`;

在 **Fragment** 中还需要额外调用 :

```
@Nullable  
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    setHasOptionsMenu(true);  
    super.onCreateView(inflater, container, savedInstanceState);  
}
```

```
<item  
    android:id="@+id/second_item_id"  
    android:orderInCategory="110"  
    android:title="@string/second_item_string"  
    android:icon="@drawable/second_item_icon"  
    app:showAsAction="ifRoom"/>
```

</menu>

For more options of configuration refer to: [Menu resource](#)

Inside **Activity**:

```
@Override  
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
    //Override defining menu resource  
    inflater.inflate(R.menu.menu_resource_id, menu);  
    super.onCreateOptionsMenu(menu, inflater);  
}
```

```
@Override  
public void onPrepareOptionsMenu(Menu menu) {  
    //Override for preparing items (setting visibility, change text, change icon...)  
    super.onPrepareOptionsMenu(menu);  
}
```

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    //Override it for handling items  
    int menuItemId = item.getItemId();  
    switch (menuItemId) {  
        case: R.id.first_item_id  
            return true; //return true, if is handled  
    }  
    return super.onOptionsItemSelected(item);  
}
```

For invoking the methods above during showing the view, call `getActivity().invalidateOptionsMenu()`;

Inside **Fragment** one additional call is needed:

```
@Nullable  
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    setHasOptionsMenu(true);  
    super.onCreateView(inflater, container, savedInstanceState);  
}
```

第39章：数据绑定库

第39.1节：基本文本字段绑定

Gradle（模块：应用）配置

```
android {  
    ...  
    dataBinding {  
        enabled = true  
    }  
}
```

数据模型

```
public class Item {  
    public String name;  
    public String description;  
  
    public Item(String name, String description) {  
        this.name = name;  
        this.description = description;  
    }  
}
```

布局 XML

第一步是将布局包裹在 `<layout>` 标签中，添加一个 `<data>` 元素，并为你的数据模型添加一个 `<variable>` 元素。

然后你可以使用 `@{model.fieldname}` 将 XML 属性绑定到数据模型中的字段，其中 `model` 是变量名，`fieldname` 是你想访问的字段。

`item_detail_activity.xml:`

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android">  
    <data>  
        <variable name="item" type="com.example.Item"/>  
    </data>  
  
    <LinearLayout  
        android:orientation="vertical"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent">  
  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@{item.name}"/>  
  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@{item.description}"/>  
  
    </LinearLayout>
```

Chapter 39: Data Binding Library

Section 39.1: Basic text field binding

Gradle (Module:app) Configuration

```
android {  
    ...  
    dataBinding {  
        enabled = true  
    }  
}
```

Data model

```
public class Item {  
    public String name;  
    public String description;  
  
    public Item(String name, String description) {  
        this.name = name;  
        this.description = description;  
    }  
}
```

Layout XML

The first step is wrapping your layout in a `<layout>` tag, adding a `<data>` element, and adding a `<variable>` element for your data model.

Then you can bind XML attributes to fields in the data model using `@{model.fieldname}`, where `model` is the variable's name and `fieldname` is the field you want to access.

`item_detail_activity.xml:`

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android">  
    <data>  
        <variable name="item" type="com.example.Item"/>  
    </data>  
  
    <LinearLayout  
        android:orientation="vertical"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent">  
  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@{item.name}"/>  
  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@{item.description}"/>  
  
    </LinearLayout>
```

```
</layout>
```

对于每个正确配置了绑定的XML布局文件，Android Gradle插件会生成一个对应的绑定类。因为我们有一个名为item_detail_activity的布局，生成的绑定类对应名为ItemDetailActivityBinding。

然后可以在Activity中这样使用该绑定：

```
public class ItemDetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ItemDetailActivityBinding binding = DataBindingUtil.setContentView(this,
            R.layout.item_detail_activity);
        Item item = new Item("示例项目", "这是一个示例项目。");
        binding.setItem(item);
    }
}
```

```
</layout>
```

For each XML layout file properly configured with bindings, the Android Gradle plugin generates a corresponding class : bindings. Because we have a layout named *item_detail_activity*, the corresponding generated binding class is called *ItemDetailActivityBinding*.

This binding can then be used in an Activity like so:

```
public class ItemDetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ItemDetailActivityBinding binding = DataBindingUtil.setContentView(this,
            R.layout.item_detail_activity);
        Item item = new Item("Example item", "This is an example item.");
        binding.setItem(item);
    }
}
```

第39.2节：内置双向数据绑定

双向数据绑定支持以下属性：

元素	属性
AbsListView	android:selectedItemPosition
CalendarView	android:date
CompoundButton	android:checked
日期选择器	• android:年 • android:月 • android:日
编辑文本框	android:text
数字选择器	android:value
单选组	android:checkedButton
评分条	android:rating
SeekBar	android:progress
TabHost	android:currentTab
TextView	android:text
TimePicker	• android:hour • android:minute
ToggleButton	android:checked
Switch	android:checked

用法

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

请注意，绑定表达式@={} 中多了一个=，这是实现双向绑定所必需的。双向绑定表达式中无法使用方法。

```
</layout>
```

For each XML layout file properly configured with bindings, the Android Gradle plugin generates a corresponding class : bindings. Because we have a layout named *item_detail_activity*, the corresponding generated binding class is called *ItemDetailActivityBinding*.

This binding can then be used in an Activity like so:

```
public class ItemDetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ItemDetailActivityBinding binding = DataBindingUtil.setContentView(this,
            R.layout.item_detail_activity);
        Item item = new Item("Example item", "This is an example item.");
        binding.setItem(item);
    }
}
```

Section 39.2: Built-in two-way Data Binding

Two-way Data-Binding supports the following attributes:

Element	Properties
AbsListView	android:selectedItemPosition
CalendarView	android:date
CompoundButton	android:checked
DatePicker	• android:year • android:month • android:day
EditText	android:text
NumberPicker	android:value
RadioGroup	android:checkedButton
RatingBar	android:rating
SeekBar	android:progress
TabHost	android:currentTab
TextView	android:text
TimePicker	• android:hour • android:minute
ToggleButton	android:checked
Switch	android:checked

Usage

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Notice that the Binding expression @={ } has an additional =, which is necessary for the **two-way Binding**. It is not possible to use methods in two-way Binding expressions.

第39.3节：使用lambda表达式的自定义事件

定义接口

```
public interface ClickHandler {  
    public void onButtonClick(User user);  
}
```

创建模型类

```
public class User {  
    private String name;  
  
    public User(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

布局 XML

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">  
  
<data>  
    <variable  
        name="handler"  
        type="com.example.ClickHandler"/>  
  
    <variable  
        name="user"  
        type="com.example.User"/>  
</data>  
  
<RelativeLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@{user.name}"  
        android:onClick="@{() -> handler.onButtonClick(user)}"/>  
    </RelativeLayout>  
</layout>
```

活动代码：

```
public class MainActivity extends Activity implements ClickHandler {  
  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {
```

Section 39.3: Custom event using lambda expression

Define Interface

```
public interface ClickHandler {  
    public void onButtonClick(User user);  
}
```

Create Model class

```
public class User {  
    private String name;  
  
    public User(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

Layout XML

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">  
  
<data>  
    <variable  
        name="handler"  
        type="com.example.ClickHandler"/>  
  
    <variable  
        name="user"  
        type="com.example.User"/>  
</data>  
  
<RelativeLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@{user.name}"  
        android:onClick="@{() -> handler.onButtonClick(user)}"/>  
    </RelativeLayout>  
</layout>
```

Activity code :

```
public class MainActivity extends Activity implements ClickHandler {  
  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
    binding.setUser(new User("DataBinding User"));
binding.setHandler(this);
}

@Override
public void onButtonClick(User user) {
Toast.makeText(MainActivity.this, "Welcome " + user.getName(),Toast.LENGTH_LONG).show();
}
}

```

对于某些在xml代码中不可用但可以在Java代码中设置的视图监听器，可以通过自定义绑定进行绑定。

自定义类

```

public class BindingUtil {
@BindingAdapter({"bind:autoAdapter"})
    public static void setAdapter(AutoCompleteTextView 视图, ArrayAdapter<String> pArrayAdapter) {
        视图.setAdapter(pArrayAdapter);
    }
@BindingAdapter({"bind:onKeyListener"})
    public static void setOnKeyListener(AutoCompleteTextView 视图 , View.OnKeyListener
pOnKeyListener)
    {
        视图.setOnKeyListener(pOnKeyListener);
    }
}

```

处理器类

```

public class 处理器 extends BaseObservable {
    private ArrayAdapter<String> 角色适配器;

    public ArrayAdapter<String> getRoleAdapter() {
        return 角色适配器;
    }
    public void setRoleAdapter(ArrayAdapter<String> pRoleAdapter) {
        角色适配器 = pRoleAdapter;
    }
}

```

XML

```

<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:bind="http://schemas.android.com/tools" >

    <data>
        <variable
            name="handler"
            type="com.example.Handler" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

```

```

super.onCreate(savedInstanceState);
binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
    binding.setUser(new User("DataBinding User"));
binding.setHandler(this);
}

@Override
public void onButtonClick(User user) {
    Toast.makeText(MainActivity.this, "Welcome " + user.getName(),Toast.LENGTH_LONG).show();
}
}

```

For some view listener which is not available in xml code but can be set in Java code, it can be bind with custom binding.

Custom class

```

public class BindingUtil {
@BindingAdapter({"bind:autoAdapter"})
    public static void setAdapter(AutoCompleteTextView view, ArrayAdapter<String> pArrayAdapter) {
        view.setAdapter(pArrayAdapter);
    }
@BindingAdapter({"bind:onKeyListener"})
    public static void setOnKeyListener(AutoCompleteTextView view , View.OnKeyListener
pOnKeyListener)
    {
        view.setOnKeyListener(pOnKeyListener);
    }
}

```

Handler class

```

public class Handler extends BaseObservable {
    private ArrayAdapter<String> roleAdapter;

    public ArrayAdapter<String> getRoleAdapter() {
        return roleAdapter;
    }
    public void setRoleAdapter(ArrayAdapter<String> pRoleAdapter) {
        roleAdapter = pRoleAdapter;
    }
}

```

XML

```

<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:bind="http://schemas.android.com/tools" >

    <data>
        <variable
            name="handler"
            type="com.example.Handler" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

```

```
<AutoCompleteTextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:singleLine="true"  
    bind:autoAdapter="@{handler.roleAdapter}" />  
  
</LinearLayout>  
</layout>
```

第39.4节：数据绑定中的默认值

预览窗格显示数据绑定表达式的默认值（如果提供）。

例如：

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

设计时将采用wrap_content，并将在预览窗格中作为wrap_content使用。

另一个例子是

```
android:text="@{user.name, default='Preview Text'}"
```

它将在预览窗格中显示Preview Text，但当你在设备/模拟器上运行时，实际绑定的文本将被显示

```
<AutoCompleteTextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:singleLine="true"  
    bind:autoAdapter="@{handler.roleAdapter}" />  
  
</LinearLayout>  
</layout>
```

Section 39.4: Default value in Data Binding

The Preview pane displays default values for data binding expressions if provided.

For example :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

It will take wrap_content while designing and will act as a wrap_content in preview pane.

Another example is

```
android:text="@{user.name, default='Preview Text'}"
```

It will display Preview Text in preview pane but when you run it in device/emulator actual text binded to it will be displayed

第39.5节：对话框中的数据绑定

```
public void doSomething() {  
    DialogTestBinding binding = DataBindingUtil.  
        .inflate(LayoutInflater.from(context), R.layout.dialog_test, null, false);  
  
    Dialog dialog = new Dialog(context);  
    dialog.setContentView(binding.getRoot());  
    dialog.show();  
}
```

Section 39.5: Databinding in Dialog

```
public void doSomething() {  
    DialogTestBinding binding = DataBindingUtil.  
        .inflate(LayoutInflater.from(context), R.layout.dialog_test, null, false);  
  
    Dialog dialog = new Dialog(context);  
    dialog.setContentView(binding.getRoot());  
    dialog.show();  
}
```

第39.6节：使用访问器方法绑定

如果你的模型有私有方法，数据绑定库仍然允许你在视图中访问它们，而无需使用方法的完整名称。

数据模型

```
public class Item {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

布局 XML

```
public class Item {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

Layout XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="item" type="com.example.Item"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!-- 由于数据模型中的"name"字段是私有的,
              此绑定将使用公共的getName()方法代替。 -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{item.name}"/>

    </LinearLayout>
</layout>

```

第39.7节：在BindingAdapter中以引用方式传递控件

布局 XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <ImageView
            android:id="@+id/img"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            app:imageUrl="@{url}"
            app:progressbar="@{progressBar}"/>

    </LinearLayout>
</layout>

```

BindingAdapter方法

```

@BindingAdapter({"imageUrl", "progressbar"})
public static void loadImage(ImageView view, String imageUrl, ProgressBar progressBar){
    Glide.with(view.getContext()).load(imageUrl)
        .listener(new RequestListener<String, GlideDrawable>() {
            @Override
            public boolean onException(Exception e, String model, Target<GlideDrawable>

```

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="item" type="com.example.Item"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!-- Since the "name" field is private on our data model,
              this binding will utilize the public getName() method instead. -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{item.name}"/>

    </LinearLayout>
</layout>

```

Section 39.7: Pass widget as reference in BindingAdapter

Layout XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <ImageView
            android:id="@+id/img"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            app:imageUrl="@{url}"
            app:progressbar="@{progressBar}"/>

    </LinearLayout>
</layout>

```

BindingAdapter method

```

@BindingAdapter({"imageUrl", "progressbar"})
public static void loadImage(ImageView view, String imageUrl, ProgressBar progressBar){
    Glide.with(view.getContext()).load(imageUrl)
        .listener(new RequestListener<String, GlideDrawable>() {
            @Override
            public boolean onException(Exception e, String model, Target<GlideDrawable>

```

```

target, boolean isFirstResource) {
    return false;
}

@Override
public boolean onResourceReady(GlideDrawable resource, String model,
Target<GlideDrawable> target, boolean isFromMemoryCache, boolean isFirstResource) {
    progressBar.setVisibility(View.GONE);
    return false;
}
}).into(view);
}

```

第39.8节：带绑定的点击监听器

为clickHandler创建接口

```

public interface ClickHandler {
    public void onButtonClick(View v);
}

```

布局 XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

<data>
    <variable
        name="handler"
        type="com.example.ClickHandler"/>
</data>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="点击我"
        android:onClick="@{handler.onButtonClick}"/>
</RelativeLayout>
</layout>

```

在你的Activity中处理事件

```

public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(View v) {
        Toast.makeText(context,"按钮被点击",Toast.LENGTH_LONG).show();
    }
}

```

```

target, boolean isFirstResource) {
    return false;
}

@Override
public boolean onResourceReady(GlideDrawable resource, String model,
Target<GlideDrawable> target, boolean isFromMemoryCache, boolean isFirstResource) {
    progressBar.setVisibility(View.GONE);
    return false;
}
}).into(view);
}

```

Section 39.8: Click listener with Binding

Create interface for clickHandler

```

public interface ClickHandler {
    public void onButtonClick(View v);
}

```

Layout XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

<data>
    <variable
        name="handler"
        type="com.example.ClickHandler"/>
</data>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="click me"
        android:onClick="@{handler.onButtonClick}"/>
</RelativeLayout>
</layout>

```

Handle event in your Activity

```

public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(View v) {
        Toast.makeText(context,"Button clicked",Toast.LENGTH_LONG).show();
    }
}

```

```
}
```

第39.9节：RecyclerView适配器中的数据绑定

也可以在你的RecyclerView适配器中使用数据绑定。

数据模型

```
public class Item {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

XML布局

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@{item.name}"/>
```

适配器类

```
public class ListItemAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {  
  
    private Activity host;  
    private List<Item> items;  
  
    public ListItemAdapter(Activity 活动, List<Item> 项目) {  
        this.host = 活动;  
        this.items = 项目;  
    }  
  
    @Override  
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        // 加载布局并获取绑定  
        ListItemBinding binding = DataBindingUtil.inflate(host.getLayoutInflater(),  
R.layout.list_item, parent, false);  
  
        return new ItemViewHolder(binding);  
    }  
  
    @Override  
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {  
        Item item = items.get(position);  
  
        ItemViewHolder itemViewHolder = (ItemViewHolder)holder;  
        itemViewHolder.bindItem(item);  
    }  
  
    @Override  
    public int getItemCount() {  
        return items.size();  
    }  
  
    private static class ItemViewHolder extends RecyclerView.ViewHolder {  
        ListItemBinding binding;
```

```
}
```

Section 39.9: Data binding in RecyclerView Adapter

It's also possible to use data binding within your RecyclerView Adapter.

Data model

```
public class Item {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

XML Layout

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@{item.name}"/>
```

Adapter class

```
public class ListItemAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {  
  
    private Activity host;  
    private List<Item> items;  
  
    public ListItemAdapter(Activity activity, List<Item> items) {  
        this.host = activity;  
        this.items = items;  
    }  
  
    @Override  
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        // inflate layout and retrieve binding  
        ListItemBinding binding = DataBindingUtil.inflate(host.getLayoutInflater(),  
R.layout.list_item, parent, false);  
  
        return new ItemViewHolder(binding);  
    }  
  
    @Override  
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {  
        Item item = items.get(position);  
  
        ItemViewHolder itemViewHolder = (ItemViewHolder)holder;  
        itemViewHolder.bindItem(item);  
    }  
  
    @Override  
    public int getItemCount() {  
        return items.size();  
    }  
  
    private static class ItemViewHolder extends RecyclerView.ViewHolder {  
        ListItemBinding binding;
```

```

ItemViewHolder(ListItemBinding binding) {
    super(binding.getRoot());
    this.binding = binding;
}

void bindItem(Item item) {
    binding.setItem(item);
    binding.executePendingBindings();
}
}

```

第39.10节：Fragment中的数据绑定

数据模型

```

public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }
}

```

布局 XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="item" type="com.example.Item"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{item.name}" />

    </LinearLayout>
</layout>

```

Fragment

```

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
 savedInstanceState) {
    FragmentTest binding = DataBindingUtil.inflate(inflater, R.layout.fragment_test, container,
        false);
    Item item = new Item();
    item.setName("Thomas");
}

```

```

ItemViewHolder(ListItemBinding binding) {
    super(binding.getRoot());
    this.binding = binding;
}

void bindItem(Item item) {
    binding.setItem(item);
    binding.executePendingBindings();
}
}

```

Section 39.10: Databinding in Fragment

Data Model

```

public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }
}

```

Layout XML

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="item" type="com.example.Item"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{item.name}" />

    </LinearLayout>
</layout>

```

Fragment

```

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
 savedInstanceState) {
    FragmentTest binding = DataBindingUtil.inflate(inflater, R.layout.fragment_test, container,
        false);
    Item item = new Item();
    item.setName("Thomas");
}

```

```
binding.setItem(item);
    return binding.getRoot();
}
```

第39.11节：使用自定义变量（整数、布尔值）的数据绑定

有时我们需要根据单个值执行基本操作，比如隐藏/显示视图，对于那个单变量我们不能创建模型，或者为此创建模型并不是好的做法。DataBinding 支持基本数据类型来执行这些操作。

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

<data>

<import type="android.view.View" />

<variable
    name="selected"
    type="Boolean" />

</data>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World"
    android:visibility="@{selected ? View.VISIBLE : View.GONE}" />

</RelativeLayout>
</layout>
```

并从 Java 类中设置其值。

```
binding.setSelected(true);
```

第39.12节：引用类

数据模型

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

布局 XML

你必须导入被引用的类，就像在 Java 中一样。

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
```

```
    binding.setItem(item);
    return binding.getRoot();
}
```

Section 39.11: DataBinding with custom variables(int,boolean)

Sometimes we need to perform basic operations like hide/show view based on single value, for that single variable we cannot create model or it is not good practice to create model for that. DataBinding supports basic datatypes to perform those operations.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

<data>

<import type="android.view.View" />

<variable
    name="selected"
    type="Boolean" />

</data>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World"
    android:visibility="@{selected ? View.VISIBLE : View.GONE}" />

</RelativeLayout>
</layout>
```

and set its value from java class.

```
binding.setSelected(true);
```

Section 39.12: Referencing classes

Data model

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

You must import referenced classes, just as you would in Java.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
```

```
<import type="android.view.View"/>
<variable name="item" type="com.example.Item"/>
</data>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- 我们引用 View 类来设置此 TextView 的可见性 -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@{item.name}"
        android:visibility="@{item.name == null ? View.VISIBLE : View.GONE}"/>

</LinearLayout>
</layout>
```

注意：包 `java.lang.*` 由系统自动导入。（JVM 对 Java 也同样处理）

```
<import type="android.view.View"/>
<variable name="item" type="com.example.Item"/>
</data>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- We reference the View class to set the visibility of this TextView -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@{item.name}"
        android:visibility="@{item.name == null ? View.VISIBLE : View.GONE}"/>

</LinearLayout>
</layout>
```

Note: The package `java.lang.*` is imported automatically by the system. (The same is made by JVM for Java)

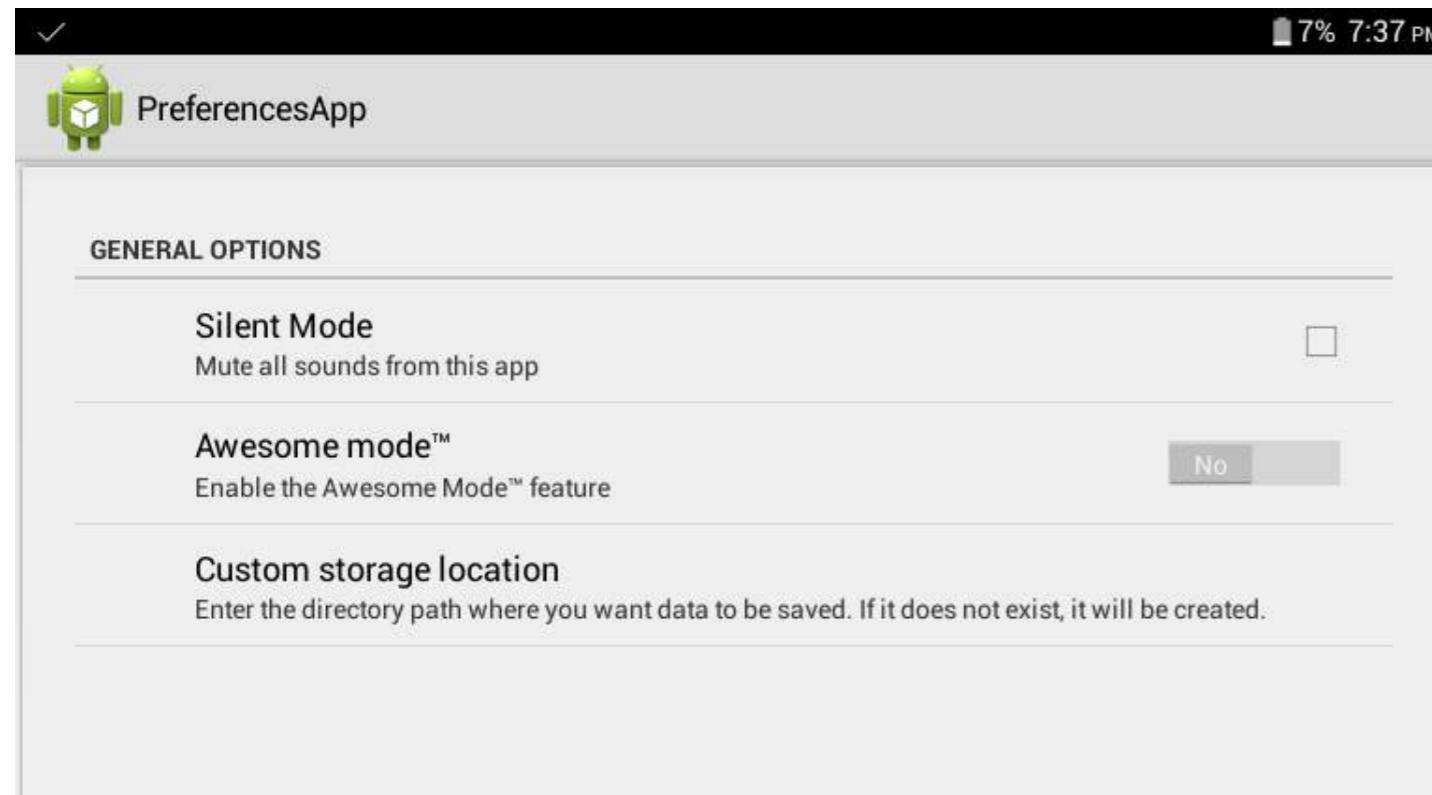
第40章：SharedPreferences

参数	详情
键	一个非空的String，用于标识参数。它可以包含空白字符或不可打印字符。此键仅在您的应用内部（以及XML文件中）使用，因此不必使用命名空间，但最好在源代码中将其作为常量。不要对其进行本地化。
默认值	所有的get函数都接受一个默认值，如果给定的键不存在，则返回该默认值。SharedPreferences。如果键存在但值类型错误，则不会返回该值：在这种情况下会抛出ClassCastException异常。

SharedPreferences 提供了一种以键值对形式将数据保存到磁盘的方法。

第40.1节：使用SharedPreferences实现设置屏幕

SharedPreferences的一个用途是在应用中实现“设置”屏幕，用户可以在其中设置他们的偏好/选项。如下所示：



PreferenceScreen 会将用户偏好保存在SharedPreferences中。要创建一个 PreferenceScreen，你需要一些东西：

用于定义可用选项的 XML 文件：

该文件放在/res/xml/preferences.xml中，对于上述设置屏幕，内容如下：

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="常规选项">
        <CheckBoxPreference
            android:key = "silent_mode"
            android:defaultValue="false"
            android:title="静音模式">
    </PreferenceCategory>
</PreferenceScreen>
```

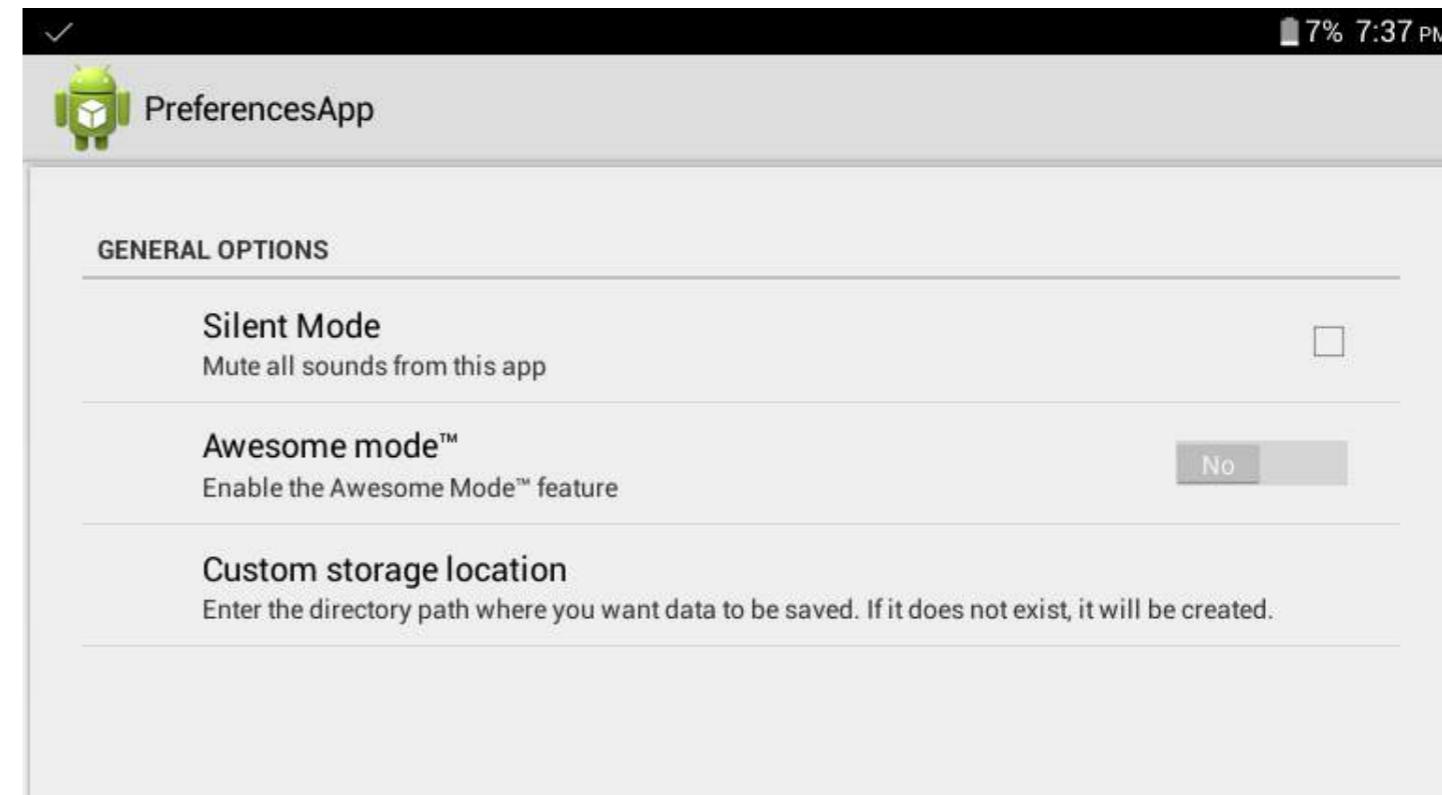
Chapter 40: SharedPreferences

Parameter	Details
key	A non-null String identifying the parameter. It can contain whitespace or non-printables. This is only used inside your app (and in the XML file), so it doesn't have to be namespaced, but it's a good idea to have it as a constant in your source code. Don't localize it.
defaultValue	All the get functions take a default value, which is returned if the given key is not present in the SharedPreferences. It's not returned if the key is present but the value has the wrong type: in that case you get a ClassCastException .

SharedPreferences provide a way to save data to disk in the form of **key-value** pairs.

Section 40.1: Implementing a Settings screen using SharedPreferences

One use of SharedPreferences is to implement a "Settings" screen in your app, where the user can set their preferences / options. Like this:



A PreferenceScreen saves user preferences in SharedPreferences. To create a PreferenceScreen, you need a few things:

An XML file to define the available options:

This goes in /res/xml/preferences.xml, and for the above settings screen, it looks like this:

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="General options">
        <CheckBoxPreference
            android:key = "silent_mode"
            android:defaultValue="false"
            android:title="Silent Mode">
    </PreferenceCategory>
</PreferenceScreen>
```

```
        android:summary="静音此应用的所有声音" />
```

```
<SwitchPreference
    android:key="awesome_mode"
    android.defaultValue="false"
    android.switchTextOn="是"
    android.switchTextOff="否"
    android:title="超赞模式™"
    android:summary="启用超赞模式™功能"/>

<EditTextPreference
    android:key="custom_storage"
    android.defaultValue="/sdcard/data/"
    android:title="自定义存储位置"
    android:summary="输入您希望保存数据的目录路径。如果该目录不存在，将会被创建。"
    android:dialogTitle="输入目录路径（例如 /sdcard/data/）"/>
</PreferenceCategory>
</PreferenceScreen>
```

这定义了设置界面中可用的选项。Android开发者文档中关于Preference类列出了许多其他类型的Preference。

接下来，我们需要一个Activity来承载我们的Preferences用户界面。在本例中，它相当简短，代码如下：

```
package com.example.preferences;

import android.preference.PreferenceActivity;
import android.os.Bundle;

public class PreferencesActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

它继承自PreferenceActivity，并提供偏好设置界面的用户界面。它可以像普通活动一样启动，在这种情况下，可以使用如下代码：

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

别忘了在你的AndroidManifest.xml中添加PreferencesActivity。

在应用内获取偏好设置的值非常简单，只需先调用setDefaultValues()，以设置XML中定义的默认值，然后获取默认的SharedPreferences。示例如下：

```
//设置我们在XML中定义的默认值
PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

//获取设置选项的值
boolean silentMode = preferences.getBoolean("silent_mode", false);
boolean awesomeMode = preferences.getBoolean("awesome_mode", false);

String customStorage = preferences.getString("custom_storage", "");
```

```
        android:summary="Mute all sounds from this app" />
```

```
<SwitchPreference
    android:key="awesome_mode"
    android.defaultValue="false"
    android.switchTextOn="Yes"
    android.switchTextOff="No"
    android:title="Awesome mode™"
    android:summary="Enable the Awesome Mode™ feature"/>
```

```
<EditTextPreference
    android:key="custom_storage"
    android.defaultValue="/sdcard/data/"
    android:title="Custom storage location"
    android:summary="Enter the directory path where you want data to be saved. If it does
not exist, it will be created."
    android:dialogTitle="Enter directory path (eg. /sdcard/data/ )"/>
</PreferenceCategory>
</PreferenceScreen>
```

This defines the available options in the settings screen. There are many other types of Preference listed in the Android Developers documentation on the [Preference Class](#).

Next, we need **an Activity to host our Preferences** user interface. In this case, it's quite short, and looks like this:

```
package com.example.preferences;

import android.preference.PreferenceActivity;
import android.os.Bundle;

public class PreferencesActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

It extends PreferenceActivity, and provides the user interface for the preferences screen. It can be started just like a normal activity, in this case, with something like:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

Don't forget to add PreferencesActivity to your `AndroidManifest.xml`.

Getting the values of the preferences inside your app is quite simple, just call `setDefaultValues()` first, in order to set the default values defined in your XML, and then get the default SharedPreferences. An example:

```
//set the default values we defined in the XML
PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

//get the values of the settings options
boolean silentMode = preferences.getBoolean("silent_mode", false);
boolean awesomeMode = preferences.getBoolean("awesome_mode", false);

String customStorage = preferences.getString("custom_storage", "");
```

第40.2节：提交 (Commit) 与应用 (Apply)

editor.apply()方法是异步的，而editor.commit()是同步的。

显然，你应该调用apply()或commit()。

版本 ≥ 2.3

```
SharedPreferences 设置 = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor 编辑器 = 设置.edit();
编辑器.putBoolean(PREF_CONST, true);
// 这将异步保存共享偏好设置，而不会阻塞当前线程。
编辑器.apply();

SharedPreferences 设置 = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor 编辑器 = 设置.edit();
编辑器.putBoolean(PREF_CONST, true);
// 这将同步保存共享偏好设置，阻塞当前线程直到完成并返回成功标志。

boolean 结果 = 编辑器.commit();
```

apply() 是在 2.3 (API 9) 中添加的，它提交时不会返回表示成功或失败的布尔值。

commit() 如果保存成功返回 true，否则返回 false。

apply() 的添加是因为 Android 开发团队注意到几乎没人关注返回值，所以 apply 更快，因为它是异步的。

与同步将偏好写入持久存储的 commit() 不同，apply() 会立即将更改提交到内存中的 SharedPreferences，但会启动异步提交到磁盘，且不会通知任何失败。如果此 SharedPreferences 上的另一个编辑器在 apply() 仍未完成时执行常规的 commit()，则 commit() 会阻塞，直到所有异步提交 (apply) 以及任何其他待处理的同步提交完成。

Section 40.2: Commit vs. Apply

The `editor.apply()` method is **asynchronous**, while `editor.commit()` is **synchronous**.

Obviously, you should call either `apply()` or `commit()`.

Version ≥ 2.3

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will asynchronously save the shared preferences without holding the current thread.
editor.apply();

SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will synchronously save the shared preferences while holding the current thread until done
and returning a success flag.
boolean result = editor.commit();
```

`apply()` was added in 2.3 (API 9), it commits without returning a boolean indicating success or failure.

`commit()` returns true if the save works, false otherwise.

`apply()` was added as the Android dev team noticed that almost no one took notice of the return value, so `apply` is faster as it is asynchronous.

Unlike `commit()`, which writes its preferences out to persistent storage synchronously, `apply()` commits its changes to the in-memory SharedPreferences immediately but starts an asynchronous commit to disk and you won't be notified of any failures. If another editor on this SharedPreferences does a regular `commit()` while a `apply()` is still outstanding, the `commit()` will block until all async commits(`apply`) are completed as well as any other sync commits that may be pending.

第40.3节：读取和写入SharedPreferences的值

```
public class MyActivity extends Activity {

    private static final String PREFS_FILE = "NameOfYourPreferenceFile";
    // PREFS_MODE 定义了哪些应用可以访问该文件
    private static final int PREFS_MODE = Context.MODE_PRIVATE;
    // 你可以使用 live template "key" 快速创建键
    private static final String KEY_BOOLEAN = "KEY_FOR_YOUR_BOOLEAN";
    private static final String KEY_STRING = "KEY_FOR_YOUR_STRING";
    private static final String KEY_FLOAT = "KEY_FOR_YOUR_FLOAT";
    private static final String KEY_INT = "KEY_FOR_YOUR_INT";
    private static final String KEY_LONG = "KEY_FOR_YOUR_LONG";

    @Override
    protected void onStart() {
        super.onStart();

        // 获取已保存的标志 (如果尚未保存则为默认值)
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        // 读取布尔值 (默认false)
        boolean booleanVal = settings.getBoolean(KEY_BOOLEAN, false);
        // 读取整型值 (默认0)
        int intValue = settings.getInt(KEY_INT, 0);
        // 读取字符串值 (默认"my string")
        String str = settings.getString(KEY_STRING, "my string");
```

Section 40.3: Read and write values to SharedPreferences

```
public class MyActivity extends Activity {

    private static final String PREFS_FILE = "NameOfYourPreferenceFile";
    // PREFS_MODE defines which apps can access the file
    private static final int PREFS_MODE = Context.MODE_PRIVATE;
    // you can use live template "key" for quickly creating keys
    private static final String KEY_BOOLEAN = "KEY_FOR_YOUR_BOOLEAN";
    private static final String KEY_STRING = "KEY_FOR_YOUR_STRING";
    private static final String KEY_FLOAT = "KEY_FOR_YOUR_FLOAT";
    private static final String KEY_INT = "KEY_FOR_YOUR_INT";
    private static final String KEY_LONG = "KEY_FOR_YOUR_LONG";

    @Override
    protected void onStart() {
        super.onStart();

        // Get the saved flag (or default value if it hasn't been saved yet)
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        // read a boolean value (default false)
        boolean booleanVal = settings.getBoolean(KEY_BOOLEAN, false);
        // read an int value (Default 0)
        int intValue = settings.getInt(KEY_INT, 0);
        // read a string value (default "my string")
        String str = settings.getString(KEY_STRING, "my string");
```

```

// 读取一个长整型值 (默认123456)
long longVal = settings.getLong(KEY_LONG, 123456);
// 读取一个浮点数值 (默认3.14f)
float floatVal = settings.getFloat(KEY_FLOAT, 3.14f);
}

@Override
protected void onStop() {
    super.onStop();

    // 保存标志
    SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
    SharedPreferences.Editor editor = settings.edit();
    // 写入一个布尔值
    editor.putBoolean(KEY_BOOLEAN, true);
    // 写入一个整型值
    editor.putInt(KEY_INT, 123);
    // 写入字符串
    editor.putString(KEY_STRING, "string value");
    // 写入长整型值
    editor.putLong(KEY_LONG, 456876451);
    // 写入浮点型值
    editor.putFloat(KEY_FLOAT, 1.51f);
    editor.apply();
}

```

`getSharedPreferences()` 是 `Context` 类中的一个方法, `Activity` 继承自该类。如果需要从其他类访问 `getSharedPreferences()` 方法, 可以使用来自 `Activity`、`View` 或 `Application` 的 `context.getSharedPreferences()` 方法。`Context` 对象引用, 来自 `Activity`、`View` 或 `Application`。

第40.4节：从特定的

`SharedPreferences` 文件中检索所有存储的条目

`getAll()` 方法检索偏好设置中的所有值。例如, 我们可以使用它来记录当前 `SharedPreferences` 的内容 :

```

private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedpreferences = context.getSharedPreferences(PREFS_FILE,
    Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedpreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ":" + value);
    }
}

```

文档警告您不要修改由 `getAll` 返回的 `Collection` :

请注意, 您不得修改此方法返回的集合, 或更改其任何内容。如果这样做, 存储数据的一致性将无法保证。

```

// read a long value (default 123456)
long longVal = settings.getLong(KEY_LONG, 123456);
// read a float value (default 3.14f)
float floatVal = settings.getFloat(KEY_FLOAT, 3.14f);
}

@Override
protected void onStop() {
    super.onStop();

    // Save the flag
    SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
    SharedPreferences.Editor editor = settings.edit();
    // write a boolean value
    editor.putBoolean(KEY_BOOLEAN, true);
    // write an integer value
    editor.putInt(KEY_INT, 123);
    // write a string
    editor.putString(KEY_STRING, "string value");
    // write a long value
    editor.putLong(KEY_LONG, 456876451);
    // write a float value
    editor.putFloat(KEY_FLOAT, 1.51f);
    editor.apply();
}

```

`getSharedPreferences()` is a method from the `Context` class — which `Activity` extends. If you need to access the `getSharedPreferences()` method from other classes, you can use `context.getSharedPreferences()` with a `Context` Object reference from an Activity, `View`, or Application.

Section 40.4: Retrieve all stored entries from a particular SharedPreferences file

The `getAll()` method retrieves all values from the preferences. We can use it, for instance, to log the current content of the `SharedPreferences`:

```

private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedpreferences = context.getSharedPreferences(PREFS_FILE,
    Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedpreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ":" + value);
    }
}

```

The documentation warns you about modifying the `Collection` returned by `getAll`:

Note that you must not modify the collection returned by this method, or alter any of its contents. The consistency of your stored data is not guaranteed if you do.

第40.5节：使用单例模式读取和写入SharedPreferences中的数据

SharedPreferences管理器（单例）类，用于读取和写入所有类型的数据。

```
import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.google.gson.Gson;

import java.lang.reflect.Type;

/**
 * 访问SharedPreferences的单例类， * 应由任何应用组件在开始时通过静态方法initialize(applicationContext)初始化一次
 */
public class SharedPrefsManager {

    private static final String TAG = SharedPrefsManager.class.getName();
    private SharedPreferences prefs;
    private static SharedPrefsManager uniqueInstance;
    public static final String PREF_NAME = "com.example.app";

    private SharedPrefsManager(Context applicationContext) {
        prefs = applicationContext.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    }

    /**
     * 如果该类未初始化，则抛出 IllegalStateException
     *
     * @return 唯一的 SharedPrefsManager 实例
     */
    public static SharedPrefsManager getInstance() {
        if (uniqueInstance == null) {
            throw new IllegalStateException(
                "SharedPrefsManager 未初始化，请先调用 initialize(applicationContext) " +
                "静态方法");
        }
        return uniqueInstance;
    }

    /**
     * 使用应用程序上下文初始化此类，
     * 应由任何应用程序组件在开始时调用一次
     *
     * @param applicationContext 应用程序上下文
     */
    public static void initialize(Context applicationContext) {
        if (applicationContext == null) {
            throw new NullPointerException("提供的应用程序上下文为null");
        }
        if (uniqueInstance == null) {
            synchronized (SharedPrefsManager.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new SharedPrefsManager(applicationContext);
                }
            }
        }
    }
}
```

Section 40.5: Reading and writing data to SharedPreferences with Singleton

SharedPreferences Manager (Singleton) class to read and write all types of data.

```
import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.google.gson.Gson;

import java.lang.reflect.Type;

/**
 * Singleton Class for accessing SharedPreferences,
 * should be initialized once in the beginning by any application component using static
 * method initialize(applicationContext)
 */
public class SharedPrefsManager {

    private static final String TAG = SharedPrefsManager.class.getName();
    private SharedPreferences prefs;
    private static SharedPrefsManager uniqueInstance;
    public static final String PREF_NAME = "com.example.app";

    private SharedPrefsManager(Context applicationContext) {
        prefs = applicationContext.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    }

    /**
     * Throws IllegalStateException if this class is not initialized
     *
     * @return unique SharedPrefsManager instance
     */
    public static SharedPrefsManager getInstance() {
        if (uniqueInstance == null) {
            throw new IllegalStateException(
                "SharedPrefsManager is not initialized, call initialize(applicationContext) " +
                "static method first");
        }
        return uniqueInstance;
    }

    /**
     * Initialize this class using application Context,
     * should be called once in the beginning by any application Component
     *
     * @param applicationContext application context
     */
    public static void initialize(Context applicationContext) {
        if (applicationContext == null) {
            throw new NullPointerException("Provided application context is null");
        }
        if (uniqueInstance == null) {
            synchronized (SharedPrefsManager.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new SharedPrefsManager(applicationContext);
                }
            }
        }
    }
}
```

```

private SharedPreferences getPrefs() {
    return prefs;
}

/**
 * 清除SharedPreferences中的所有数据
 */
public void clearPrefs() {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.clear();
    editor.commit();
}

public void removeKey(String key) {
    getPrefs().edit().remove(key).commit();
}

public boolean containsKey(String key) {
    return getPrefs().contains(key);
}

public String getString(String key, String defaultValue) {
    return getPrefs().getString(key, defaultValue);
}

public String getString(String key) {
    return getString(key, null);
}

public void setString(String key, String value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putString(key, value);
    editor.apply();
}

public int getInt(String key, int defaultValue) {
    return getPrefs().getInt(key, defaultValue);
}

public int getInt(String key) {
    return getInt(key, 0);
}

public void setInt(String key, int value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putInt(key, value);
    editor.apply();
}

public long getLong(String key, long defaultValue) {
    return getPrefs().getLong(key, defaultValue);
}

public long getLong(String key) {
    return getLong(key, 0L);
}

public void setLong(String key, long value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putLong(key, value);
    editor.apply();
}

```

```

private SharedPreferences getPrefs() {
    return prefs;
}

/**
 * Clears all data in SharedPreferences
 */
public void clearPrefs() {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.clear();
    editor.commit();
}

public void removeKey(String key) {
    getPrefs().edit().remove(key).commit();
}

public boolean containsKey(String key) {
    return getPrefs().contains(key);
}

public String getString(String key, String defaultValue) {
    return getPrefs().getString(key, defaultValue);
}

public String getString(String key) {
    return getString(key, null);
}

public void setString(String key, String value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putString(key, value);
    editor.apply();
}

public int getInt(String key, int defaultValue) {
    return getPrefs().getInt(key, defaultValue);
}

public int getInt(String key) {
    return getInt(key, 0);
}

public void setInt(String key, int value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putInt(key, value);
    editor.apply();
}

public long getLong(String key, long defaultValue) {
    return getPrefs().getLong(key, defaultValue);
}

public long getLong(String key) {
    return getLong(key, 0L);
}

public void setLong(String key, long value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putLong(key, value);
    editor.apply();
}

```

```

public boolean getBoolean(String key, boolean defValue) {
    return getPrefs().getBoolean(key, defValue);
}

public boolean getBoolean(String key) {
    return getBoolean(key, false);
}

public void setBoolean(String key, boolean value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putBoolean(key, value);
editor.apply();
}

public boolean getFloat(String key) {
    return getFloat(key, 0f);
}

public boolean getFloat(String key, float defValue) {
    return getFloat(key, defValue);
}

public void setFloat(String key, Float value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putFloat(key, value);
editor.apply();
}

```

/* 在指定的键中将对象持久化到偏好设置中，给定对象的类必须实现 Model

*** 接口**

*** @param key 字符串 ***
@param modelObject 要持久化的对象 * @par
am <M> 对象的泛型

```

public <M extends Model> void set0bject(String key, M model0bject) {
    String value = createJSONStringFrom0bject(model0bject);
SharedPreferences.Editor editor = getPrefs().edit();
    editor.putString(key, value);
editor.apply();
}

```

/*

*** 从偏好设置中获取之前存储的给定类的对象**

*** @param key 字符串**
*** @param classOfModelObject 持久化对象的类**
*** @param <M> 对象的泛型**

*** @return 给定类的对象**
***/**

```

public <M extends Model> M get0bject(String key, Class<M> classOfModel0bject) {
    String jsonData = getPrefs().getString(key, null);
    if (null != jsonData) {
        try {
Gson gson = new Gson();
        M customObject = gson.fromJson(jsonData, classOfModel0bject);
        return customObject;
    } 捕获 (ClassCastException cce) {
Log.d(TAG, "无法将从偏好设置中获取的字符串转换为类型为 " +
+ cce.getMessage());
    }
classOfModelObject.getName() + ""

```

```

public boolean getBoolean(String key, boolean defValue) {
    return getPrefs().getBoolean(key, defValue);
}

public boolean getBoolean(String key) {
    return getBoolean(key, false);
}

public void setBoolean(String key, boolean value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putBoolean(key, value);
editor.apply();
}

public boolean getFloat(String key) {
    return getFloat(key, 0f);
}

public boolean getFloat(String key, float defValue) {
    return getFloat(key, defValue);
}

public void setFloat(String key, Float value) {
    SharedPreferences.Editor editor = getPrefs().edit();
    editor.putFloat(key, value);
editor.apply();
}

/**
* Persists an Object in prefs at the specified key, class of given Object must implement Model
* interface
*
* @param key      String
* @param modelObject Object to persist
* @param <M>      Generic for Object
*/
public <M extends Model> void set0bject(String key, M model0bject) {
    String value = createJSONStringFrom0bject(model0bject);
SharedPreferences.Editor editor = getPrefs().edit();
    editor.putString(key, value);
editor.apply();
}

/**
* Fetches the previously stored Object of given Class from prefs
*
* @param key      String
* @param classOfModelObject Class of persisted Object
* @param <M>      Generic for Object
* @return Object of given class
*/
public <M extends Model> M get0bject(String key, Class<M> classOfModel0bject) {
    String jsonData = getPrefs().getString(key, null);
    if (null != jsonData) {
        try {
Gson gson = new Gson();
        M customObject = gson.fromJson(jsonData, classOfModel0bject);
        return customObject;
    } catch (ClassCastException cce) {
Log.d(TAG, "Cannot convert string obtained from prefs into collection of type " +
classOfModel0bject.getName() + "\n" + cce.getMessage());
    }
}

```

```

    }

    return null;
}

/**
* 在指定的键中持久化一个 Collection 对象
*
* @param key      字符串
* @param dataCollection Collection 对象
* @param <C>      Collection 对象的泛型
*/
public <C> void setCollection(String key, C dataCollection) {
    SharedPreferences.Editor editor = getPrefs().edit();
    String value = createJSONStringFromObject(dataCollection);
    editor.putString(key, value);
    editor.apply();
}

/**
* 从偏好设置中获取之前存储的指定类型的 Collection 对象
*
* @param key      字符串
* @param typeOfC Collection 对象的类型
* @param <C>      Collection 对象的泛型
* @return 可被强制转换的 Collection 对象
*/
public <C> C getCollection(String key, Type typeOfC) {
    String jsonData = getPrefs().getString(key, null);
    if (null != jsonData) {
        try {
Gson gson = new Gson();
        C arrFromPrefs = gson.fromJson(jsonData, typeOfC);
        return arrFromPrefs;
    } 捕获 (ClassCastException cce) {
Log.d(TAG, "无法将从偏好设置中获取的字符串转换为类型为 " +
message());
    }
}
return null;
}

public void registerPrefsListener(SharedPreferences.OnSharedPreferenceChangeListener listener)
{
getPrefs().registerOnSharedPreferenceChangeListener(listener);
}

public void unregisterPrefsListener()

SharedPreferences.OnSharedPreferenceChangeListener listener) {
    getPrefs().unregisterOnSharedPreferenceChangeListener(listener);
}

public SharedPreferences.Editor getEditor() {
    return getPrefs().edit();
}

private static String createJSONStringFromObject(Object object) {
    Gson gson = new Gson();
    return gson.toJson(object);
}

```

```

    }

    return null;
}

/**
* Persists a Collection object in prefs at the specified key
*
* @param key      String
* @param dataCollection Collection Object
* @param <C>      Generic for Collection object
*/
public <C> void setCollection(String key, C dataCollection) {
    SharedPreferences.Editor editor = getPrefs().edit();
    String value = createJSONStringFromObject(dataCollection);
    editor.putString(key, value);
    editor.apply();
}

/**
* Fetches the previously stored Collection Object of given type from prefs
*
* @param key      String
* @param typeOfC Type of Collection Object
* @param <C>      Generic for Collection Object
* @return Collection Object which can be casted
*/
public <C> C getCollection(String key, Type typeOfC) {
    String jsonData = getPrefs().getString(key, null);
    if (null != jsonData) {
        try {
Gson gson = new Gson();
        C arrFromPrefs = gson.fromJson(jsonData, typeOfC);
        return arrFromPrefs;
    } catch (ClassCastException cce) {
Log.d(TAG, "Cannot convert string obtained from prefs into collection of type " +
typeOfC.toString() + "\n" + cce.getMessage());
    }
}
return null;
}

public void registerPrefsListener(SharedPreferences.OnSharedPreferenceChangeListener listener)
{
    getPrefs().registerOnSharedPreferenceChangeListener(listener);
}

public void unregisterPrefsListener()

    SharedPreferences.OnSharedPreferenceChangeListener listener) {
    getPrefs().unregisterOnSharedPreferenceChangeListener(listener);
}

public SharedPreferences.Editor getEditor() {
    return getPrefs().edit();
}

private static String createJSONStringFromObject(Object object) {
    Gson gson = new Gson();
    return gson.toJson(object);
}

```

Model interface which is implemented by classes going to Gson to avoid proguard obfuscation.

```
public interface Model {  
}
```

Proguard 规则针对Model接口：

```
-keep interface com.example.app.Model  
-keep class * implements com.example.app.Model { *;}
```

第40.6节：getPreferences(int) 与 getSharedPreferences(String, int) 的比较

getPreferences(int)

返回由Activity的类名保存的偏好设置，如文档中所述：

检索一个 SharedPreferences 对象，用于访问该活动私有的偏好设置。此方法简单地调用底层的 getSharedPreferences(String, int) 方法，传入该活动的类名作为偏好设置名称。

而使用 `getSharedPreferences (String name, int mode)` 方法则返回保存在给定name下的偏好设置。如文档中所述：

检索并持有名为 'name' 的偏好设置文件内容，返回一个 SharedPreferences，通过它可以检索和修改其值。

因此，如果保存在SharedPreferences中的值需要在整个应用中使用，应使用 `getSharedPreferences (String name, int mode)` 并指定固定名称。因为使用 `getPreferences (int)` 返回/保存的是属于调用该方法的Activity的偏好设置。

第40.7节：监听SharedPreferences的变化

```
SharedPreferences sharedPreferences = ...;  
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);
```

```
private final SharedPreferences.OnSharedPreferenceChangeListener mOnSharedPreferenceChangeListener  
= new SharedPreferences.OnSharedPreferenceChangeListener() {  
    @Override  
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {  
        //TODO  
    }  
}
```

请注意：

- 监听器只有在值被添加或更改时才会触发，设置相同的值不会调用它；
- 监听器需要保存在成员变量中，而不能使用匿名类，因为 `registerOnSharedPreferenceChangeListener` 会以弱引用存储它，因此它会被垃圾回收；

Model interface which is implemented by classes going to Gson to avoid proguard obfuscation.

```
public interface Model {  
}
```

Proguard rules for Model interface:

```
-keep interface com.example.app.Model  
-keep class * implements com.example.app.Model { *;}
```

Section 40.6: getPreferences(int) VS getSharedPreferences(String, int)

getPreferences([int](#))

returns the preferences saved by Activity's [class name](#) as described in the [docs](#) :

Retrieve a SharedPreferences object for accessing preferences that are private to this activity. This simply calls the underlying getSharedPreferences(String, int) method by passing in this activity's class name as the preferences name.

While using `getSharedPreferences (String name, int mode)` method returns the prefs saved under the given name. As in the docs :

Retrieve and hold the contents of the preferences file 'name', returning a SharedPreferences through which you can retrieve and modify its values.

So if the value being saved in the SharedPreferences has to be used across the app, one should use `getSharedPreferences (String name, int mode)` with a fixed name. As, using `getPreferences(int)` returns/saves the preferences belonging to the Activity calling it.

Section 40.7: Listening for SharedPreferences changes

```
SharedPreferences sharedPreferences = ...;  
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);
```

```
private final SharedPreferences.OnSharedPreferenceChangeListener mOnSharedPreferenceChangeListener  
= new SharedPreferences.OnSharedPreferenceChangeListener() {  
    @Override  
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {  
        //TODO  
    }  
}
```

Please note:

- The listener will fire only if value was added or changed, setting the same value won't call it;
- The listener needs to be saved in a member variable and **NOT** with an anonymous class, because `registerOnSharedPreferenceChangeListener` stores it with a weak reference, so it would be garbage collected;

- 除了使用成员变量外，也可以由类直接实现，然后调用 `registerOnSharedPreferenceChangeListener(this);`
- 当不再需要时，记得注销监听器，使用 `unregisterSharedPreferenceChangeListener` 更改监听器。

第40.8节：从

`SharedPreferences` 存储、检索、移除和清除数据

创建 `SharedPreferences` `BuyyaPref`

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref", MODE_PRIVATE);
Editor editor = pref.edit();
```

以键/值对形式存储数据

```
editor.putBoolean("key_name1", true);           // 保存布尔值 - true/false
editor.putInt("key_name2", 10);                 // 保存整数
editor.putFloat("key_name3", 10.1f);             // 保存浮点数
editor.putLong("key_name4", 1000);               // 保存长整型
editor.putString("key_name5", "MyString");       // 保存字符串

// 在 SharedPreferences 中保存更改
editor.commit(); // 提交更改
```

获取 `SharedPreferences` 数据

如果键对应的值不存在，则返回第二个参数的值（本例中为 `null`，类似默认值）

```
pref.getBoolean("key_name1", null);           // 获取布尔值
pref.getInt("key_name2", null);                // 获取整数
pref.getFloat("key_name3", null);              // 获取浮点数
pref.getLong("key_name4", null);               // 获取长整型
pref.getString("key_name5", null);             // 获取字符串
```

从 `SharedPreferences` 中删除键值

```
editor.remove("key_name3"); // 将删除键 key_name3
editor.remove("key_name4"); // 将删除键 key_name4

// 保存 SharedPreferences 中的更改
editor.commit(); // 提交更改
```

清除 `SharedPreferences` 中的所有数据

```
editor.clear();
editor.commit(); // 提交更改
```

第40.9节：为 `EditTextPreference` 添加过滤器

创建此类：

```
public class InputFilterMinMax implements InputFilter {

    private int min, max;

    public InputFilterMinMax(int min, int max) {
```

- Instead of using a member variable, it can also be directly implemented by the class and then call `registerOnSharedPreferenceChangeListener(this);`
- Remember to unregister the listener when it is no more required using `unregisterOnSharedPreferenceChangeListener`.

Section 40.8: Store, Retrieve, Remove and Clear Data from Shared Preferences

Create `SharedPreferences` `BuyyaPref`

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref", MODE_PRIVATE);
Editor editor = pref.edit();
```

Storing data as KEY/VALUE pair

```
editor.putBoolean("key_name1", true);           // Saving boolean - true/false
editor.putInt("key_name2", 10);                 // Saving integer
editor.putFloat("key_name3", 10.1f);             // Saving float
editor.putLong("key_name4", 1000);               // Saving long
editor.putString("key_name5", "MyString");       // Saving string

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Get `SharedPreferences` data

If value for key not exist then return second param value (In this case `null`, this is like default value)

```
pref.getBoolean("key_name1", null);           // getting boolean
pref.getInt("key_name2", null);                // getting Integer
pref.getFloat("key_name3", null);              // getting Float
pref.getLong("key_name4", null);               // getting Long
pref.getString("key_name5", null);             // getting String
```

Deleting Key value from `SharedPreferences`

```
editor.remove("key_name3"); // will delete key key_name3
editor.remove("key_name4"); // will delete key key_name4

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Clear all data from `SharedPreferences`

```
editor.clear();
editor.commit(); // commit changes
```

Section 40.9: Add filter for `EditTextPreference`

Create this class :

```
public class InputFilterMinMax implements InputFilter {

    private int min, max;

    public InputFilterMinMax(int min, int max) {
```

```

        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int dstart,
    int dend) {
        try {
            int input = Integer.parseInt(dest.toString() + source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}

```

用法：

```

EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });

```

第40.10节：SharedPreferences中支持的数据类型

SharedPreferences 只允许存储原始数据类型 (boolean、float、long、int、String 和 stringset)。你不能在 SharedPreferences 中存储更复杂的对象，因此它实际上是用来存储用户设置或类似内容的地方，而不是用作保存用户数据的数据库（例如保存用户制作的待办事项列表）。

要在 SharedPreferences 中存储内容，你需要使用一个键 (Key) 和值 (Value)。键是你以后引用所存储内容的方式，值是你想要存储的数据。

```

String keyToUseToFindLater = "High Score";
int newHighScore = 12938;
//获取 SharedPreferences 和 Editor 对象
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
//在 SharedPreferences 文件中保存一个 int 值
editor.putInt(keyToUseToFindLater, newHighScore);
editor.commit();

```

第40.11节：实例化 SharedPreferences 对象的不同方式

你可以通过多种方式访问 SharedPreferences：

获取默认的SharedPreferences文件：

```

        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int dstart,
    int dend) {
        try {
            int input = Integer.parseInt(dest.toString() + source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}

```

用法：

```

EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });

```

Section 40.10: Supported data types in SharedPreferences

SharedPreferences allows you to store primitive data types only (**boolean, float, long, int, String, and string set**). You cannot store more complex objects in SharedPreferences, and as such is really meant to be a place to store user settings or similar, it's not meant to be a database to keep user data (like saving a todo list a user made for example).

To store something in SharedPreferences you use a Key and a Value. The Key is how you can reference what you stored later and the Value data you want to store.

```

String keyToUseToFindLater = "High Score";
int newHighScore = 12938;
//getting SharedPreferences & Editor objects
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
//saving an int in the SharedPreferences file
editor.putInt(keyToUseToFindLater, newHighScore);
editor.commit();

```

Section 40.11: Different ways of instantiating an object of SharedPreferences

You can access SharedPreferences in several ways:

Get the default SharedPreferences file:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

获取特定的 SharedPreferences 文件：

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

从另一个应用获取 SharedPreferences：

```
// 注意另一个应用必须将 prefs 声明为 MODE_WORLD_WRITEABLE
final ArrayList<HashMap<String, String>> LIST = new ArrayList<HashMap<String, String>>();
Context contextOtherApp = createPackageContext("com.otherapp", Context.MODE_WORLD_WRITEABLE);
SharedPreferences prefs = contextOtherApp.getSharedPreferences("pref_file_name",
Context.MODE_WORLD_READABLE);
```

第40.12节：移除键

```
private static final String MY_PREF = "MyPref";

// ...

SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

在apply()之后，prefs包含了"key" -> "value"，以及它之前已经包含的内容。虽然看起来我先添加了"key"然后又移除了它，但实际上移除操作先发生。Editor中的所有更改是一次性应用的，而不是按照你添加它们的顺序。所有的移除操作都发生在所有的添加操作之前。

第40.13节：支持带有StringSet的Honeycomb之前版本

这是工具类：

```
public class SharedPreferencesCompat {

    public static void putStringSet(SharedPreferences.Editor editor, String key, Set<String> values) {
        if (Build.VERSION.SDK_INT >= 11) {
            while (true) {
                try {
                    editor.putStringSet(key, values).apply();
                    break;
                } 捕获 (ClassCastException ex) {
                    // 清除系统升级前的陈旧JSON字符串
                    editor.remove(key);
                }
            }
        } else putStringSetToJson(editor, key, values);
    }

    public static Set<String> getStringSet(SharedPreferences prefs, String key, Set<String> defaultValue)
    {
```

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Get a specific SharedPreferences file:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Get SharedPreferences from another app:

```
// Note that the other app must declare prefs as MODE_WORLD_WRITEABLE
final ArrayList<HashMap<String, String>> LIST = new ArrayList<HashMap<String, String>>();
Context contextOtherApp = createPackageContext("com.otherapp", Context.MODE_WORLD_WRITEABLE);
SharedPreferences prefs = contextOtherApp.getSharedPreferences("pref_file_name",
Context.MODE_WORLD_READABLE);
```

Section 40.12: Removing keys

```
private static final String MY_PREF = "MyPref";

// ...

SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

After the apply(), prefs contains "key" -> "value", in addition to whatever it contained already. Even though it looks like I added "key" and then removed it, the remove actually happens first. The changes in the Editor are all applied in one go, not in the order you added them. All removes happen before all puts.

Section 40.13: Support pre-Honeycomb with StringSet

Here's the utility class:

```
public class SharedPreferencesCompat {

    public static void putStringSet(SharedPreferences.Editor editor, String key, Set<String> values) {
        if (Build.VERSION.SDK_INT >= 11) {
            while (true) {
                try {
                    editor.putStringSet(key, values).apply();
                    break;
                } catch (ClassCastException ex) {
                    // Clear stale JSON string from before system upgrade
                    editor.remove(key);
                }
            }
        } else putStringSetToJson(editor, key, values);
    }

    public static Set<String> getStringSet(SharedPreferences prefs, String key, Set<String> defaultValue) {
```

```

if (Build.VERSION.SDK_INT >= 11) {
    try {
        return prefs.getStringSet(key, defaultReturnValue);
    } catch (ClassCastException ex) {
        // 如果用户从Gingerbread升级到更高版本，读取陈旧的JSON字符串
        return getStringSetFromJson(prefs, key, defaultReturnValue);
    }
} else return getStringSetFromJson(prefs, key, defaultReturnValue);
}

private static Set<String> getStringSetFromJson(SharedPreferences prefs, String key,
Set<String> defaultReturnValue) {
    final String input = prefs.getString(key, null);
    if (input == null) return defaultReturnValue;

    try {
        HashSet<String> set = new HashSet<>();
        JSONArray json = new JSONArray(input);
        for (int i = 0, size = json.length(); i < size; i++) {
            String value = json.getString(i);
            set.add(value);
        }
        return set;
    } catch (JSONException e) {
        e.printStackTrace();
        return defaultReturnValue;
    }
}

private static void putStringSetToJson(SharedPreferences.Editor editor, String key, Set<String> values) {
    JSONArray json = new JSONArray(values);
    if (Build.VERSION.SDK_INT >= 9)
        editor.putString(key, json.toString()).apply();
    else
        editor.putString(key, json.toString()).commit();
}

private SharedPreferencesCompat() {}
}

```

保存偏好设置为StringSet数据类型的示例是：

```

Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);

```

要取回它们：

```

Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());

```

[参考：Android Support Preference](#)

```

if (Build.VERSION.SDK_INT >= 11) {
    try {
        return prefs.getStringSet(key, defaultReturnValue);
    } catch (ClassCastException ex) {
        // If user upgraded from Gingerbread to something higher read the stale JSON string
        return getStringSetFromJson(prefs, key, defaultReturnValue);
    }
} else return getStringSetFromJson(prefs, key, defaultReturnValue);
}

private static Set<String> getStringSetFromJson(SharedPreferences prefs, String key,
Set<String> defaultReturnValue) {
    final String input = prefs.getString(key, null);
    if (input == null) return defaultReturnValue;

    try {
        HashSet<String> set = new HashSet<>();
        JSONArray json = new JSONArray(input);
        for (int i = 0, size = json.length(); i < size; i++) {
            String value = json.getString(i);
            set.add(value);
        }
        return set;
    } catch (JSONException e) {
        e.printStackTrace();
        return defaultReturnValue;
    }
}

private static void putStringSetToJson(SharedPreferences.Editor editor, String key, Set<String> values) {
    JSONArray json = new JSONArray(values);
    if (Build.VERSION.SDK_INT >= 9)
        editor.putString(key, json.toString()).apply();
    else
        editor.putString(key, json.toString()).commit();
}

private SharedPreferencesCompat() {}
}

```

An example to save preferences as StringSet data type is:

```

Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);

```

To retrieve them back:

```

Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());

```

[Reference: Android Support Preference](#)

第41章：Intent

参数	详细信息
意图	启动意图
requestCode	用于识别请求的唯一编号
options	启动Activity时的附加选项
name	额外数据的名称
value	额外数据的值
CHOOSE_CONTACT_REQUEST_CODE	请求代码，用于在 <code>onActivityResult</code> 方法中识别该请求
action	通过此意图执行的任何操作，例如： <code>Intent.ACTION_VIEW</code>
uri	意图用于执行指定操作的数据uri
packageContext	用于初始化Intent的上下文
cls	此意图使用的类

Intent是Android系统中传递的小消息。该消息可能包含我们执行任务的意图信息。

它基本上是一个被动的数据结构，包含要执行操作的抽象描述。

第41.1节：从另一个Activity获取结果

通过使用`startActivityForResult(Intent intent, int requestCode)`你可以启动另一个Activity，然后在`onActivityResult(int requestCode, int resultCode, Intent data)`方法中接收该Activity的结果。结果将作为一个Intent返回。Intent可以通过Bundle携带数据

在这个例子中，MainActivity将启动一个DetailActivity并期待其返回结果。每种请求类型应有自己唯一的int请求码，这样在MainActivity中重写的`onActivityResult(int requestCode, int resultCode, Intent data)`方法里，可以通过比较requestCode和REQUEST_CODE_EXAMPLE的值来确定处理哪个请求（尽管在此例中只有一个请求码）。

MainActivity :

```
public class MainActivity extends Activity {

    // 为每个用例使用唯一的请求码
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // 创建一个新的Intent实例以启动DetailActivity
        final Intent intent = new Intent(this, DetailActivity.class);

        // 使用请求码启动DetailActivity
        startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
    }

    // 只有在onActivityResult被调用时
    // 当另一个活动之前使用 startActivityForResult 启动时
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Chapter 41: Intent

Parameter	Details
intent	The intent to start
requestCode	Unique number to identify the request
options	Additional options for how the Activity should be started
name	The name of the extra data
value	The value of the extra data
CHOOSE_CONTACT_REQUEST_CODE	the code of the request, to identify it on <code>onActivityResult</code> method
action	Any action to perform via this intent, ex: <code>Intent.ACTION_VIEW</code>
uri	data uri to be used by intent to perform specified action
packageContext	Context to use to initialize the Intent
cls	Class to be used by this intent

An Intent is a small message passed around the Android system. This message may hold information about our intention to perform a task.

It is basically a passive data structure holding an abstract description of an action to be performed.

Section 41.1: Getting a result from another Activity

By using `startActivityForResult(Intent intent, int requestCode)` you can start another `Activity` and then receive a result from that `Activity` in the `onActivityResult(int requestCode, int resultCode, Intent data)` method. The result will be returned as an `Intent`. An intent can contain data via a `Bundle`

In this example MainActivity will start a DetailActivity and then expect a result from it. Each request type should have its own `int` request code, so that in the overridden `onActivityResult(int requestCode, int resultCode, Intent data)` method in MainActivity , it can be determined which request to process by comparing values of requestCode and REQUEST_CODE_EXAMPLE (though in this example, there is only one).

MainActivity:

```
public class MainActivity extends Activity {

    // Use a unique request code for each use case
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Create a new instance of Intent to start DetailActivity
        final Intent intent = new Intent(this, DetailActivity.class);

        // Start DetailActivity with the request code
        startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
    }

    // onActivityResult only get called
    // when the other Activity previously started using startActivityForResult
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

```

// 首先我们需要检查 requestCode 是否与我们使用的一致。
if(requestCode == REQUEST_CODE_EXAMPLE) {

    // resultCode 由 DetailActivity 设置// 按惯例, RESULT_OK 表示 DetailActivity 执行成功
    if(resultCode == Activity.RESULT_OK) {
        // 从返回的 Intent 中获取结果
        final String result = data.getStringExtra(DetailActivity.EXTRA_DATA);

        // 使用数据——在此例中, 将其显示为 Toast。
        Toast.makeText(this, "结果: " + result, Toast.LENGTH_LONG).show();
    } else {
        // setResult 未被 DetailActivity 成功执行
        // 由于某些错误或控制流程。无数据可检索。
    }
}

}

```

DetailActivity:

```

public class DetailActivity extends Activity {

    // 用于识别活动间传递数据的常量。
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        final Button button = (Button) findViewById(R.id.button);
        // 当此按钮被点击时, 我们希望返回一个结果
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // 创建一个新的 Intent 对象作为结果的容器
                final Intent data = new Intent();

                // 将所需数据添加到 MainActivity 返回
                data.putExtra(EXTRA_DATA, "一些有趣的数据!");

                // 将 requestCode 设置为 Activity.RESULT_OK// 表示成功
                // , 并附加包含结果数据的 Intent
                setResult(Activity.RESULT_OK, data);

                // 使用 finish() 关闭 DetailActivity
                // 返回到 MainActivity
                finish();
            }
        });
    }

    @Override
    public void onBackPressed() {
        // 当用户按下返回键时, 将 requestCode 设置为
        // Activity.RESULT_CANCELED 表示失败
        setResult(Activity.RESULT_CANCELED);
        super.onBackPressed();
    }
}

```

```

// First we need to check if the requestCode matches the one we used.
if(requestCode == REQUEST_CODE_EXAMPLE) {

    // The resultCode is set by the DetailActivity
    // By convention RESULT_OK means that whatever
    // DetailActivity did was executed successfully
    if(resultCode == Activity.RESULT_OK) {
        // Get the result from the returned Intent
        final String result = data.getStringExtra(DetailActivity.EXTRA_DATA);

        // Use the data - in this case, display it in a Toast.
        Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
    } else {
        // setResult wasn't successfully executed by DetailActivity
        // Due to some error or flow of control. No data to retrieve.
    }
}

```

DetailActivity:

```

public class DetailActivity extends Activity {

    // Constant used to identify data sent between Activities.
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        final Button button = (Button) findViewById(R.id.button);
        // When this button is clicked we want to return a result
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Create a new Intent object as container for the result
                final Intent data = new Intent();

                // Add the required data to be returned to the MainActivity
                data.putExtra(EXTRA_DATA, "Some interesting data!");

                // Set the resultCode as Activity.RESULT_OK to
                // indicate a success and attach the Intent
                // which contains our result data
                setResult(Activity.RESULT_OK, data);

                // With finish() we close the DetailActivity to
                // return back to MainActivity
                finish();
            }
        });
    }

    @Override
    public void onBackPressed() {
        // When the user hits the back button set the resultCode
        // as Activity.RESULT_CANCELED to indicate a failure
        setResult(Activity.RESULT_CANCELED);
        super.onBackPressed();
    }
}

```

}

你需要注意的几点：

- 只有调用 `finish()` 后才会返回数据。你需要在调用 `finish()` 之前调用 `setResult()`，否则不会返回任何结果。
- 确保你的 `Activity` 没有使用 `android:launchMode="singleTask"`，否则会导致 `Activity` 在单独的任务中运行，因此你将无法接收到它的结果。如果你的 `Activity` 使用 `singleTask` 作为启动模式，它会立即调用 `onActivityResult()`，且结果码为 `Activity.RESULT_CANCELED`。
- 使用 `android:launchMode="singleInstance"` 时要小心。在 Lollipop (Android 5.0, API 等级 21) 之前的设备上，`Activity` 不会返回结果。
- 您可以在调用 `startActivityForResult()` 时使用显式(explicit)或隐式(implicit)意图。当启动您自己的活动以接收结果时，您应使用显式意图以确保收到预期的结果。显式意图(intent)总是传递给其目标，无论其内容如何；不会检查过滤器(filter)。但隐式意图仅在能够通过组件的某个过滤器时才传递给该组件。

}

A few things you need to be aware of:

- Data is only returned once you call `finish()`. You need to call `setResult()` before calling `finish()`, otherwise, no result will be returned.
- Make sure your `Activity` is not using `android:launchMode="singleTask"`, or it will cause the `Activity` to run in a separate task and therefore you will not receive a result from it. If your `Activity` uses `singleTask` as launch mode, it will call `onActivityResult()` immediately with a result code of `Activity.RESULT_CANCELED`.
- Be careful when using `android:launchMode="singleInstance"`. On devices before Lollipop (Android 5.0, API Level 21), `Activities` will not return a result.
- You can use `explicit` or `implicit` intents when you call `startActivityForResult()`. When starting one of your own activities to receive a result, you should use an explicit intent to ensure that you receive the expected result. An explicit intent is always delivered to its target, no matter what it contains; the filter is not consulted. But an implicit intent is delivered to a component only if it can pass through one of the component's filters.

第41.2节：活动之间传递数据

此示例演示如何将值为“Some data!”的String从OriginActivity发送到DestinationActivity。

注意：这是在两个活动之间发送数据的最直接方式。有关更健壮实现的示例，请参见使用启动者模式的示例。

OriginActivity

```
public class OriginActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        // 创建一个新的Intent对象，目标活动为DestinationActivity。
        final Intent intent = new Intent(this, DestinationActivity.class);

        // 使用putExtra()以键/值对的形式向intent对象添加数据
        intent.putExtra(DestinationActivity.EXTRA_DATA, "Some data!");

        // 使用intent对象启动目标活动
        startActivity(intent);
    }
}
```

DestinationActivity

```
public class DestinationActivity extends AppCompatActivity {
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_destination);

        // getIntent() 返回用于启动此 Activity 的 Intent 对象
        final Intent intent = getIntent();
    }
}
```

Section 41.2: Passing data between activities

This example illustrates sending a `String` with value as “`Some data!`” from `OriginActivity` to `DestinationActivity`.

NOTE: This is the most straightforward way of sending data between two activities. See the example on using the starter pattern for a more robust implementation.

OriginActivity

```
public class OriginActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        // Create a new Intent object, containing DestinationActivity as target Activity.
        final Intent intent = new Intent(this, DestinationActivity.class);

        // Add data in the form of key/value pairs to the intent object by using putExtra()
        intent.putExtra(DestinationActivity.EXTRA_DATA, "Some data!");

        // Start the target Activity with the intent object
        startActivity(intent);
    }
}
```

DestinationActivity

```
public class DestinationActivity extends AppCompatActivity {
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_destination);

        // getIntent() returns the Intent object which was used to start this Activity
        final Intent intent = getIntent();
    }
}
```

```
// 通过使用与 OriginActivity 中添加数据时相同的键，从 intent 对象中检索数据
// ...
final String data = intent.getStringExtra(EXTRA_DATA);
}
```

也可以传递其他基本类型数据以及数组、[Bundle](#)和[Parcelable](#)数据。传递 [Serializable](#) 也是可能的，但应避免使用，因为它比Parcelable慢三倍以上。

Serializable是一个标准的 Java接口。你只需通过实现

[Serializable](#) 接口来标记一个类为Serializable，Java 会在需要时自动序列化它。

[Parcelable](#)是 Android 特有的接口，可以在自定义数据类型（即你自己的对象/ POJO 对象）上实现，它允许你的对象被扁平化并自行重建，而目标端无需做任何操作。文档中有一个制作对象可 Parcelable 的示例。

一旦你有了一个parcelable对象，你就可以像发送原始类型一样通过intent对象发送它：

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

或者放在bundle中／作为fragment的参数：

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

然后也可以在目标处通过getParcelableExtra从intent中读取它：

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

或者在fragment中从bundle读取时：

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

一旦你有了一个Serializable对象，你就可以将它放入intent对象中：

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

然后在目标处从意图对象中读取它，如下所示：

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

第41.3节：在浏览器中打开URL

使用默认浏览器打开

此示例演示如何通过编程方式在内置网页浏览器中打开URL，而不是在您的应用程序内打开。这允许您的应用打开网页，而无需在清单文件中包含INTERNET权限。

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // 验证该意图是否能解析到一个活动
    if (intent.resolveActivity(getApplicationContext()) != null) {
        // 这里我们使用一个没有选择器的意图，与下一个示例不同
    }
}
```

```
// Retrieve the data from the intent object by using the same key that
// was previously used to add data to the intent object in OriginActivity.
final String data = intent.getStringExtra(EXTRA_DATA);
}
```

It is also possible to pass other primitive data types as well as arrays, [Bundle](#) and [Parcelable](#) data. Passing [Serializable](#) is also possible, but should be avoided as it is more than three times slower than Parcelable.

Serializable is a standard Java [interface](#). You simply mark a class as [Serializable](#) by implementing the [Serializable interface](#) and Java will automatically serialize it during required situations.

Parcelable is an Android specific [interface](#) which can be implemented on custom data types (i.e. your own objects / POJO objects), it allows your object to be flattened and reconstruct itself without the destination needing to do anything. There is a documentation example of making an object parcelable.

Once you have a parcelable object you can send it like a primitive type, with an intent object:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

Or in a bundle / as an argument for a fragment:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

and then also read it from the intent at the destination using getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

Or when reading in a fragment from a bundle:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Once you have a [Serializable](#) object you can put it in an intent object:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

and then also read it from the intent object at the destination as shown below:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Section 41.3: Open a URL in a browser

Opening with the default browser

This example shows how you can open a URL programmatically in the built-in web browser rather than within your application. This allows your app to open up a webpage without the need to include the INTERNET permission in your manifest file.

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // Verify that the intent will resolve to an activity
    if (intent.resolveActivity(getApplicationContext()) != null) {
        // Here we use an intent without a Chooser unlike the next example
    }
}
```

```
startActivity(intent);
}
}
```

提示用户选择浏览器

请注意，此示例使用了Intent.createChooser()方法：

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    // 注意下面的选择器。如果没有应用程序匹配,
    // Android 会显示系统消息。因此这里不需要 try-catch。
    startActivity(Intent.createChooser(intent, "使用浏览器打开"));
}
```

在某些情况下，URL 可能以"www"开头。如果是这种情况，你会遇到以下异常：

```
android.content.ActivityNotFoundException : 找不到处理该 Intent 的 Activity
```

URL 必须始终以"http://"或"https://"开头。因此你的代码应检查这一点，如下面的代码片段所示：

```
if (!url.startsWith("https://") && !url.startsWith("http://")){
    url = "http://" + url;
}
Intent openUrlIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
if (openUrlIntent.resolveActivity(getApplicationContext()) != null) {
    startActivity(openUrlIntent);
}
```

最佳实践

检查设备上是否有应用可以接收隐式 Intent。否则，当你的应用调用startActivity()时会崩溃。要先验证是否存在应用可以接收该 Intent，调用 Intent 对象的resolveActivity()方法。如果结果非空，说明至少有一个应用可以处理该 Intent，调用startActivity()是安全的。如果结果为空，则不应使用该 Intent，并且如果可能，应禁用调用该 Intent 的功能。

第41.4节：启动器模式

这种模式是一种更严格的启动Activity的方法。其目的是提高代码的可读性，同时降低代码复杂性、维护成本以及组件之间的耦合度。

以下示例实现了启动者模式，通常作为Activity自身的静态方法实现。

该静态方法接受所有必需的参数，从这些数据构造一个有效的Intent，然后启动该Activity。

Intent是一个在运行时绑定不同组件（例如两个活动）之间的对象。Intent表示应用程序“想要做某事”的意图。你可以使用Intent完成各种任务，但这里你的意图是启动另一个活动。

```
public class ExampleActivity extends AppCompatActivity {
```

```
    startActivity(intent);
}
}
```

Prompting the user to select a browser

Note that this example uses the [Intent.createChooser\(\)](#) method:

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    // Note the Chooser below. If no applications match,
    // Android displays a system message. So here there is no need for try-catch.
    startActivity(Intent.createChooser(intent, "Browse with"));
}
```

In some cases, the URL may start with "www". If that is the case you will get this exception:

```
android.content.ActivityNotFoundException : No Activity found to handle Intent
```

The URL must always start with "**http://**" or "**https://**". Your code should therefore check for it, as shown in the following code snippet:

```
if (!url.startsWith("https://") && !url.startsWith("http://")){
    url = "http://" + url;
}
Intent openUrlIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
if (openUrlIntent.resolveActivity(getApplicationContext()) != null) {
    startActivity(openUrlIntent);
}
```

Best Practices

Check if there are no apps on the device that can receive the implicit intent. Otherwise, your app will crash when it calls `startActivity()`. To first verify that an app exists to receive the intent, call `resolveActivity()` on your Intent object. If the result is non-null, there is at least one app that can handle the intent and it's safe to call `startActivity()`. If the result is null, you should not use the intent and, if possible, you should disable the feature that invokes the intent.

Section 41.4: Starter Pattern

This pattern is a more strict approach to starting an Activity. Its purpose is to improve code readability, while at the same time decrease code complexity, maintenance costs, and coupling of your components.

The following example implements the starter pattern, which is usually implemented as a static method on the Activity itself. This static method accepts all required parameters, constructs a valid Intent from that data, and then starts the Activity.

An Intent is an object that provides runtime binding between separate components, such as two activities. The Intent represents an app's "intent to do something." You can use intents for a wide variety of tasks, but here, your intent starts another activity.

```
public class ExampleActivity extends AppCompatActivity {
```

```

private static final String EXTRA_DATA = "EXTRA_DATA";

public static void start(Context context, String data) {
    Intent intent = new Intent(context, ExampleActivity.class);
    intent.putExtra(EXTRA_DATA, data);
    context.startActivity(intent);
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent intent = getIntent();
    if(!intent.getExtras().containsKey(EXTRA_DATA)){
        throw new UnsupportedOperationException("Activity 应该使用静态启动方法启动");
    }
    String data = intent.getStringExtra(EXTRA_DATA);
}
}

```

该模式还允许您强制通过意图传递额外的数据。

然后可以像这样启动ExampleActivity，其中context是一个活动上下文：

```
ExampleActivity.start(context, "Some data!");
```

第41.5节：清除活动栈

有时您可能希望启动一个新活动，同时移除返回栈中的先前活动，这样返回按钮就不会带您回到它们。一个例子是启动应用时进入登录活动，随后进入应用的主活动，但在注销时希望直接返回登录界面且无法返回。在这种情况下，您可以为意图设置FLAG_ACTIVITY_CLEAR_TOP标志，意味着如果要启动的活动已经在当前任务中运行（LoginActivity），那么不会启动该活动的新实例，而是关闭其上方的所有其他活动，并将此意图作为新意图传递给（现在位于顶部的）旧活动。

```

Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);

```

如果您想清除返回栈上的所有活动，也可以将FLAG_ACTIVITY_NEW_TASK与FLAG_ACTIVITY_CLEAR_TASK标志一起使用：

```

Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
// 关闭所有活动，清除返回栈。
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);

```

第41.6节：启动活动

这个示例将从OriginActivity启动DestinationActivity。

这里，Intent 构造函数接受两个参数：

1. 一个 Context 作为第一个参数（因为 Activity 类是 Context 的子类，所以使用它）

```

private static final String EXTRA_DATA = "EXTRA_DATA";

public static void start(Context context, String data) {
    Intent intent = new Intent(context, ExampleActivity.class);
    intent.putExtra(EXTRA_DATA, data);
    context.startActivity(intent);
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent intent = getIntent();
    if(!intent.getExtras().containsKey(EXTRA_DATA)){
        throw new UnsupportedOperationException("Activity should be started using the static start method");
    }
    String data = intent.getStringExtra(EXTRA_DATA);
}
}

```

This pattern also allows you to force additional data to be passed with the intent.

The ExampleActivity can then be started like this, where context is an activity context:

```
ExampleActivity.start(context, "Some data!");
```

Section 41.5: Clearing an activity stack

Sometimes you may want to start a new activity while removing previous activities from the back stack, so the back button doesn't take you back to them. One example of this might be starting an app on the Login activity, taking you through to the Main activity of your application, but on logging out you want to be directed back to Login without a chance to go back. In a case like that you can set the FLAG_ACTIVITY_CLEAR_TOP flag for the intent, meaning if the activity being launched is already running in the current task (LoginActivity), then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) old activity as a new Intent.

```

Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);

```

It's also possible to use the flags FLAG_ACTIVITY_NEW_TASK along with FLAG_ACTIVITY_CLEAR_TASK if you want to clear all Activities on the back stack:

```

Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
// Closing all the Activities, clear the back stack.
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);

```

Section 41.6: Start an activity

This example will start DestinationActivity from OriginActivity.

Here, the Intent constructor takes two parameters:

1. A Context as its first parameter (this is used because the Activity class is a subclass of Context)

2. 系统应将 Intent 传递给的应用组件的类（在本例中，是应该启动的活动）

```
public class OriginActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_origin);  
  
        Intent intent = new Intent(this, DestinationActivity.class);  
  
        startActivity(intent);  
    }  
}  
  
finish(); // 可选地，你可以关闭 OriginActivity。这样当用户从 DestinationActivity 按返回键时，他/她不会再次回到 OriginActivity。
```

创建打开DestinationActivity的Intent的另一种方法是使用Intent的默认构造函数，并使用setClass()方法告诉它要打开哪个 Activity：

```
Intent i=new Intent();  
i.setClass(this, DestinationActivity.class);  
startActivity(intent);  
finish(); // 可选地，你可以关闭 OriginActivity。这样当用户从 DestinationActivity 按返回键时，他/她不会回到 OriginActivity
```

第41.7节：发送电子邮件

```
// 使用 'mailto' 方案编译一个 Uri  
Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(  
    "mailto","johndoe@example.com", null));  
// 主题  
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello World!");  
// 邮件正文  
emailIntent.putExtra(Intent.EXTRA_TEXT, "Hi! I am sending you a test email.");  
// 文件附件  
emailIntent.putExtra(Intent.EXTRA_STREAM, attachedFileUri);  
  
// 检查设备是否有电子邮件客户端  
if (emailIntent.resolveActivity(getApplicationContext()) != null) {  
    // 提示用户选择邮件应用  
    startActivityForResult(Intent.createChooser(emailIntent,"选择您的邮件应用"));  
} else {  
    // 通知用户未安装邮件客户端或提供替代方案  
}
```

这将预先填写用户选择的邮件应用中的电子邮件。

如果需要添加附件，可以使用Intent.ACTION_SEND代替Intent.ACTION_SENDTO。对于多个附件，可以使用ACTION_SEND_MULTIPLE

注意：并非所有设备都支持ACTION_SENDTO，且在未先通过resolveActivity()检查的情况下调用startActivity()可能会抛出ActivityNotFoundException异常。

第41.8节：用于Chrome自定义标签页的CustomTabsIntent

版本 ≥ 4.0.3

2. The Class of the app component to which the system should deliver the Intent (in this case, the activity that should be started)

```
public class OriginActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_origin);  
  
        Intent intent = new Intent(this, DestinationActivity.class);  
  
        startActivity(intent);  
    }  
}  
  
finish(); // Optionally, you can close OriginActivity. In this way when the user press back from DestinationActivity he/she won't land on OriginActivity again.
```

Another way to create the Intent to open DestinationActivity is to use the default constructor for the Intent, and use the setClass() method to tell it which Activity to open:

```
Intent i=new Intent();  
i.setClass(this, DestinationActivity.class);  
startActivity(intent);  
finish(); // Optionally, you can close OriginActivity. In this way when the user press back from DestinationActivity he/she won't land on OriginActivity
```

Section 41.7: Sending emails

```
// Compile a Uri with the 'mailto' schema  
Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(  
    "mailto","johndoe@example.com", null));  
// Subject  
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello World!");  
// Body of email  
emailIntent.putExtra(Intent.EXTRA_TEXT, "Hi! I am sending you a test email.");  
// File attachment  
emailIntent.putExtra(Intent.EXTRA_STREAM, attachedFileUri);  
  
// Check if the device has an email client  
if (emailIntent.resolveActivity(getApplicationContext()) != null) {  
    // Prompt the user to select a mail app  
    startActivityForResult(Intent.createChooser(emailIntent,"Choose your mail application"));  
} else {  
    // Inform the user that no email clients are installed or provide an alternative  
}
```

This will pre-fill an email in a mail app of the user's choice.

If you need to add an attachment, you can use Intent.ACTION_SEND instead of Intent.ACTION_SENDTO. For multiple attachments you can use ACTION_SEND_MULTIPLE

A word of caution: not every device has a provider for ACTION_SENDTO, and calling startActivityForResult() without checking with [resolveActivity\(\)](#) first may throw an ActivityNotFoundException.

Section 41.8: CustomTabsIntent for Chrome Custom Tabs

Version ≥ 4.0.3

使用CustomTabsIntent，现在可以配置Chrome自定义标签页，以自定义从您的应用打开的浏览器中的关键UI组件。

这是某些情况下使用WebView的一个很好的替代方案。它允许通过Intent加载网页，并且可以在浏览器中注入一定程度的应用外观和感觉。

以下是如何使用CustomTabsIntent打开URL的示例

```
String url = "https://www.google.pl/";
CustomTabsIntent intent = new CustomTabsIntent.Builder()
    .setStartAnimations(getApplicationContext(), R.anim.slide_in_right, R.anim.slide_out_left)
    .setExitAnimations(getApplicationContext(), android.R.anim.slide_in_left,
        android.R.anim.slide_out_right)
.setCloseButtonIcon(BitmapFactory.decodeResource(getResources(),
    R.drawable.ic_arrow_back_white_24dp))
    .setToolbarColor(Color.parseColor("#43A047"))
    .enableUrlBarHiding()
.build();
intent.launchUrl(getActivity(), Uri.parse(url));
```

注意：

要使用自定义标签页，您需要在build.gradle中添加此依赖项

```
编译 'com.android.support:customtabs:24.1.1'
```

第41.9节：Intent URI

本示例展示了如何从浏览器启动intent：

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">启动
intent</a>
```

此intent将启动包名为com.sample.test的应用，或打开该包对应的Google Play页面。

此intent也可以通过JavaScript启动：

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

在activity中，可以从intent数据中获取此host和path：

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // 返回 host.com/path
}
```

Intent URI语法：

HOST/URI-路径 // 可选主机
#Intent;
 package=[字符串];
 action=[字符串];
 category=[字符串];
 component=[字符串];
 scheme=[字符串];

Using a [CustomTabsIntent](#), it is now possible to configure [Chrome custom tabs](#) in order to customize key UI components in the browser that is opened from your app.

This is a good alternative to using a WebView for some cases. It allows loading of a web page with an Intent, with the added ability to inject some degree of the look and feel of your app into the browser.

Here is an example of how to open a url using CustomTabsIntent

```
String url = "https://www.google.pl/";
CustomTabsIntent intent = new CustomTabsIntent.Builder()
    .setStartAnimations(getApplicationContext(), R.anim.slide_in_right, R.anim.slide_out_left)
    .setExitAnimations(getApplicationContext(), android.R.anim.slide_in_left,
        android.R.anim.slide_out_right)
.setCloseButtonIcon(BitmapFactory.decodeResource(getResources(),
    R.drawable.ic_arrow_back_white_24dp))
    .setToolbarColor(Color.parseColor("#43A047"))
    .enableUrlBarHiding()
.build();
intent.launchUrl(getActivity(), Uri.parse(url));
```

Note:

To use custom tabs, you need to add this dependency to your build.gradle

```
compile 'com.android.support:customtabs:24.1.1'
```

Section 41.9: Intent URI

This example shows, how to start intent from browser:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

This intent will start app with package com.sample.test or will open google play with this package.

Also this intent can be started with javascript:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

In activity this host and path can be obtained from intent data:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Intent URI syntax:

HOST/URI-path // Optional host
#Intent;
 package=[string];
 action=[string];
 category=[string];
 component=[string];
 scheme=[string];

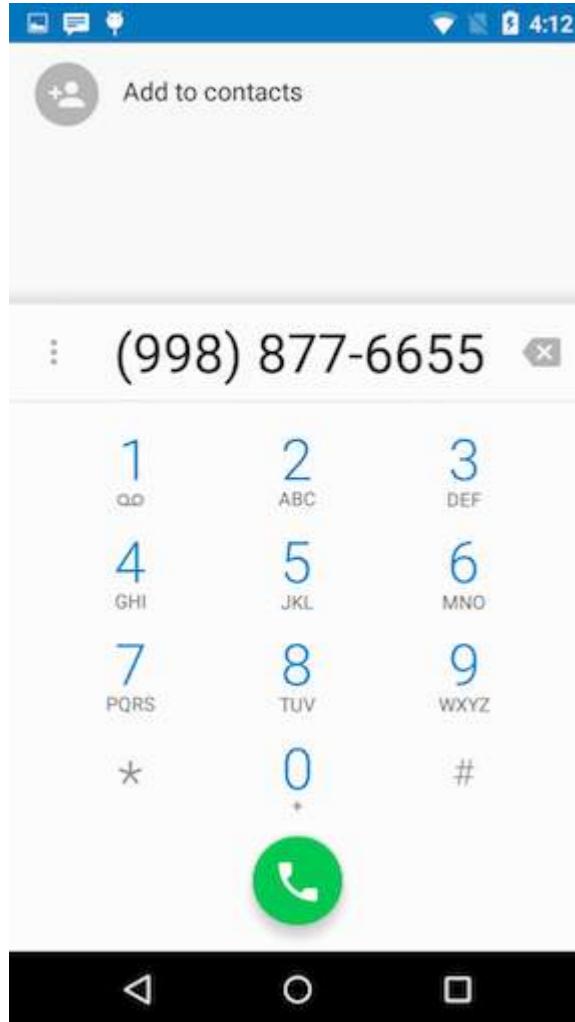
end;

第41.10节：启动拨号器

此示例展示了如何打开默认拨号器（一个用于拨打常规电话的应用），并预先填入电话号码：

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:9988776655")); //替换为有效电话号码。记得添加tel:前缀，否则会崩溃。
startActivity(intent);
```

运行上述代码的结果：



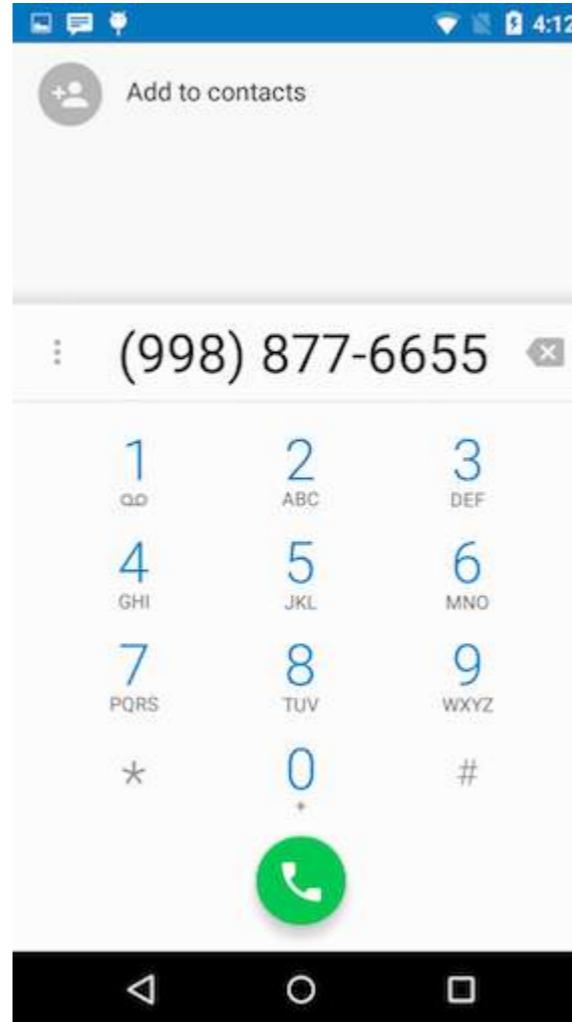
end;

Section 41.10: Start the dialer

This example shows how to open a default dialer (an app that makes regular calls) with a provided telephone number already in place:

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:9988776655")); //Replace with valid phone number. Remember to add the tel: prefix, otherwise it will crash.
startActivity(intent);
```

Result from running the code above:



第41.11节：向其他组件广播消息

Intent可用于向应用的其他组件（例如正在运行的后台服务）或整个Android系统广播消息。

要在您的应用程序内发送广播，请使用LocalBroadcastManager类：

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // 意图动作
intent.putExtra("key", "value"); // 通过广播传递的数据
```

```
LocalBroadcastManager manager = LocalBroadcastManager.getInstance(context);
manager.sendBroadcast(intent);
```

Section 41.11: Broadcasting Messages to Other Components

Intents can be used to broadcast messages to other components of your application (such as a running background service) or to the entire Android system.

To send a broadcast **within your application**, use the LocalBroadcastManager class:

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast
```

```
LocalBroadcastManager manager = LocalBroadcastManager.getInstance(context);
manager.sendBroadcast(intent);
```

要向应用程序外的组件发送广播，请使用Context

对象的 sendBroadcast() 方法。

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // 意图动作  
intent.putExtra("key", "value"); // 通过广播传递的数据  
  
context.sendBroadcast(intent);
```

有关接收广播的信息，请参见：Broadcast Receiver

第41.12节：在活动之间传递自定义对象

也可以使用Bundle类将自定义对象传递给其他活动。

有两种方式：

- Serializable接口—适用于Java和Android
- Parcelable接口—内存高效，仅适用于Android（推荐）

Parcelable

Parcelable的处理速度远快于Serializable。原因之一是我们明确指定了序列化过程，而不是使用反射来推断它。还有一个合理的推断是，代码已针对这一目的进行了大量优化。

```
public class MyObjects implements Parcelable {  
  
    private int age;  
    private String name;  
  
    private ArrayList<String> address;  
  
    public MyObjects(String name, int age, ArrayList<String> address) {  
        this.name = name;  
        this.age = age;  
        this.address = address;  
    }  
  
    public MyObjects(Parcel source) {  
        age = source.readInt();  
        name = source.readString();  
        address = source.createStringArrayList();  
    }  
  
    @Override  
    public int describeContents() {  
        return 0;  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags) {  
        dest.writeInt(age);  
        dest.writeString(name);  
        dest.writeStringList(address);  
    }  
  
    public int getAge() {  
        return age;  
    }
```

To send a broadcast to components outside of your application, use the sendBroadcast() method on a Context object.

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action  
intent.putExtra("key", "value"); // data to be passed with your broadcast  
  
context.sendBroadcast(intent);
```

Information about *receiving* broadcasts can be found here: Broadcast Receiver

Section 41.12: Passing custom object between activities

It is also possible to pass your custom object to other activities using the [Bundle](#) class.

There are two ways:

- Serializable interface—for Java and Android
- Parcelable interface—memory efficient, only for Android (recommended)

Parcelable

Parcelable processing is much faster than serializable. One of the reasons for this is that we are explicit about the serialization process instead of using reflection to infer it. It also stands to reason that the code has been heavily optimized for this purpose.

```
public class MyObjects implements Parcelable {  
  
    private int age;  
    private String name;  
  
    private ArrayList<String> address;  
  
    public MyObjects(String name, int age, ArrayList<String> address) {  
        this.name = name;  
        this.age = age;  
        this.address = address;  
    }  
  
    public MyObjects(Parcel source) {  
        age = source.readInt();  
        name = source.readString();  
        address = source.createStringArrayList();  
    }  
  
    @Override  
    public int describeContents() {  
        return 0;  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags) {  
        dest.writeInt(age);  
        dest.writeString(name);  
        dest.writeStringList(address);  
    }  
  
    public int getAge() {  
        return age;  
    }
```

```

}

public String getName() {
    return name;
}

public ArrayList<String> getAddress() {
    if (!(address == null))
        return address;
    否则
        return new ArrayList<String>();
}

public static final Creator<MyObjects> CREATOR = new Creator<MyObjects>() {
    @Override
    public MyObjects[] newArray(int size) {
        return new MyObjects[size];
    }

    @Override
    public MyObjects createFromParcel(Parcel source) {
        return new MyObjects(source);
    }
};
}

```

发送活动代码

```

MyObject mObject = new MyObject("姓名", "年龄", "地址数组");

//传递 MyObject
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putExtra("UniqueKey", mObject);
startActivity(mIntent);

```

在目标活动中接收对象。

```

//获取 MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");

```

你可以如下传递 Parcelable 对象的 ArrayList

```

//MyObjects 数组
ArrayList<MyObject> mUsers;

//传递 MyObject 列表
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putParcelableArrayListExtra("UniqueKey", mUsers);
startActivity(mIntent);

//获取 MyObject 列表
Intent mIntent = getIntent();
ArrayList<MyObjects> mUsers = mIntent.getParcelableArrayList("UniqueKey");

```

注意：有一些 Android Studio 插件，比如这个，可以用来生成 Parcelable 代码

Serializable

```

}

public String getName() {
    return name;
}

public ArrayList<String> getAddress() {
    if (!(address == null))
        return address;
    else
        return new ArrayList<String>();
}

public static final Creator<MyObjects> CREATOR = new Creator<MyObjects>() {
    @Override
    public MyObjects[] newArray(int size) {
        return new MyObjects[size];
    }

    @Override
    public MyObjects createFromParcel(Parcel source) {
        return new MyObjects(source);
    }
};
}

```

Sending Activity Code

```

MyObject mObject = new MyObject("name", "age", "Address array here");

//Passing MyObject
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putExtra("UniqueKey", mObject);
startActivity(mIntent);

```

Receiving the object in destination activity.

```

//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");

```

You can pass ArrayList of Parcelable object as below

```

//Array of MyObjects
ArrayList<MyObject> mUsers;

//Passing MyObject List
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putParcelableArrayListExtra("UniqueKey", mUsers);
startActivity(mIntent);

//Getting MyObject List
Intent mIntent = getIntent();
ArrayList<MyObjects> mUsers = mIntent.getParcelableArrayList("UniqueKey");

```

Note: There are Android Studio plugins such as [this one](#) available to generate Parcelable code

Serializable

发送活动代码

```
Product product = new Product();
Bundle bundle = new Bundle();
bundle.putSerializable("product", product);
Intent cartIntent = new Intent(mContext, ShowCartActivity.class);
cartIntent.putExtras(bundle);
mContext.startActivity(cartIntent);
```

在目标活动中接收对象。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
}
```

Serializable 对象的 ArrayList：与单个对象传递相同

自定义对象应实现 [Serializable](#) 接口。

第41.13节：使用指定的纬度和经度打开谷歌地图

您可以通过 Intent 将纬度和经度从您的应用传递给谷歌地图

```
String uri = String.format(Locale.ENGLISH, "http://maps.google.com/maps?q=loc:%f,%f",
28.43242324,77.8977673);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
startActivity(intent);
```

第41.14节：通过Activity中的Intent传递不同的数据

1. 传递整数数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("intVariableName", intValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
int intValue = mIntent.getIntExtra("intVariableName", 0); // 如果未找到 intVariableName 的值，则设置默认值为 0
```

2. 传递双精度数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("doubleVariableName", doubleValue);
startActivity(myIntent);
```

Sending Activity Code

```
Product product = new Product();
Bundle bundle = new Bundle();
bundle.putSerializable("product", product);
Intent cartIntent = new Intent(mContext, ShowCartActivity.class);
cartIntent.putExtras(bundle);
mContext.startActivity(cartIntent);
```

Receiving the object in destination activity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
}
```

ArrayList of Serializable object: same as single object passing

Custom object should implement the [Serializable](#) interface.

Section 41.13: Open Google map with specified latitude, longitude

You can pass latitude, longitude from your app to Google map using Intent

```
String uri = String.format(Locale.ENGLISH, "http://maps.google.com/maps?q=loc:%f,%f",
28.43242324,77.8977673);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
startActivity(intent);
```

Section 41.14: Passing different data through Intent in Activity

1. Passing integer data:

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("intVariableName", intValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
int intValue = mIntent.getIntExtra("intVariableName", 0); // set 0 as the default value if no value
for intVariableName found
```

2. Passing double data:

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("doubleVariableName", doubleValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
double doubleValue = mIntent.getDoubleExtra("doubleVariableName", 0.00); // 如果未找到 doubleVariableName 的值，则设置默认值为 0.00
```

3. 传递字符串数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("stringVariableName", stringValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
String stringValue = mIntent.getExtras().getString("stringVariableName");
```

或

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. 传递 ArrayList 数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5. 传递对象数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

注意：请记住你的自定义类必须实现Serializable接口。

6. 传递 HashMap<String, String> 数据：

ReceiverActivity

```
Intent mIntent = getIntent();
double doubleValue = mIntent.getDoubleExtra("doubleVariableName", 0.00); // set 0.00 as the default value if no value for doubleVariableName found
```

3. Passing String data:

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("stringVariableName", stringValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
String stringValue = mIntent.getExtras().getString("stringVariableName");
```

or

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. Passing ArrayList data :

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5. Passing Object data :

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

Note : Keep in mind your custom Class must implement the [Serializable](#) interface.

6. Passing HashMap<String, String> data :

SenderIdActivity

```
HashMap<String, String> hashMap;

Intent mIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
mIntent.putExtra("hashMap", hashMap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
HashMap<String, String> hashMap = (HashMap<String, String>)
mIntent.getSerializableExtra("hashMap");
```

7. 传递位图数据：

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image", bitmap);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

第41.15节：共享意图

与不同应用共享简单信息。

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "这是我要发送的文本。");
sendIntent.setType("text/plain");
startActivity(Intent.createChooser(sendIntent, getResources().getText(R.string.send_to)));
```

与不同应用共享图片。

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

第41.16节：显示文件选择器并读取结果

启动文件选择器活动

```
public void showFileChooser() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);

    // 更新MIME类型
    intent.setType("*/*");

    // 在这里使用String[]更新额外的MIME类型。
    intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes);
```

SenderIdActivity

```
HashMap<String, String> hashMap;
```

```
Intent mIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
mIntent.putExtra("hashMap", hashMap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
HashMap<String, String> hashMap = (HashMap<String, String>)
mIntent.getSerializableExtra("hashMap");
```

7. Passing Bitmap data :

SenderIdActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image", bitmap);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Section 41.15: Share intent

Share simple information with different apps.

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(Intent.createChooser(sendIntent, getResources().getText(R.string.send_to)));
```

Share an image with different apps.

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Section 41.16: Showing a File Chooser and Reading the Result

Starting a File Chooser Activity

```
public void showFileChooser() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);

    // Update with mime types
    intent.setType("*/*");

    // Update with additional mime types here using a String[].
    intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes);
```

```

// 仅选择可打开且本地的文件。理论上我们可以从Google云端硬盘
// 或其他具有网络文件的应用程序中提取文件，但对于本示例来说这没有必要。
intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.putExtra(Intent.EXTRA_LOCAL_ONLY, true);

// REQUEST_CODE = <some-integer>
startActivityForResult(intent, REQUEST_CODE);
}

```

读取结果

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // 如果用户没有选择文件则直接返回
    if (requestCode != REQUEST_CODE || resultCode != RESULT_OK) {
        return;
    }

    // 导入文件
    importFile(data.getData());
}

public void importFile(Uri uri) {
    String fileName = getFileName(uri);

    // 临时文件可以是你想要的任何文件
    File fileCopy = copyToTempFile(uri, File tempFile)

    // 完成！
}

/**
 * 使用内容解析器获取URI的文件名。摘自以下链接
 * https://developer.android.com/training/secure-file-sharing/retrieve-info.html#RetrieveFileInfo
 *
 * @param uri 要查询的 URI
 * @return 不带路径的文件名
 * @throws IllegalArgumentException 如果查询为 null、空或列不存在
 */
private String getFileName(Uri uri) throws IllegalArgumentException {
    // 获取有关此 URI 的游标信息
    Cursor cursor = getContentResolver().query(uri, null, null, null, null);

    if (cursor.getCount() <= 0) {
        cursor.close();
        throw new IllegalArgumentException("无法获取文件名，游标为空");
    }

    cursor.moveToFirst();

    String fileName = cursor.getString(cursor.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME));

    cursor.close();

    return fileName;
}

/**
 * 将 URI 引用复制到临时文件
 *
 * @param uri      用作输入流的 URI
 * @param tempFile 用作输出流的文件

```

```

// Only pick openable and local files. Theoretically we could pull files from google drive
// or other applications that have networked files, but that's unnecessary for this example.
intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.putExtra(Intent.EXTRA_LOCAL_ONLY, true);

// REQUEST_CODE = <some-integer>
startActivityForResult(intent, REQUEST_CODE);
}

Reading the Result

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the user doesn't pick a file just return
    if (requestCode != REQUEST_CODE || resultCode != RESULT_OK) {
        return;
    }

    // Import the file
    importFile(data.getData());
}

public void importFile(Uri uri) {
    String fileName = getFileName(uri);

    // The temp file could be whatever you want
    File fileCopy = copyToTempFile(uri, File tempFile)

    // Done!
}

/**
 * Obtains the file name for a URI using content resolvers. Taken from the following link
 * https://developer.android.com/training/secure-file-sharing/retrieve-info.html#RetrieveFileInfo
 *
 * @param uri a uri to query
 * @return the file name with no path
 * @throws IllegalArgumentException if the query is null, empty, or the column doesn't exist
 */
private String getFileName(Uri uri) throws IllegalArgumentException {
    // Obtain a cursor with information regarding this uri
    Cursor cursor = getContentResolver().query(uri, null, null, null, null);

    if (cursor.getCount() <= 0) {
        cursor.close();
        throw new IllegalArgumentException("Can't obtain file name, cursor is empty");
    }

    cursor.moveToFirst();

    String fileName = cursor.getString(cursor.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME));

    cursor.close();

    return fileName;
}

/**
 * Copies a uri reference to a temporary file
 *
 * @param uri      the uri used as the input stream
 * @param tempFile the file used as an output stream

```

```

* @return 方便起见返回输入的 tempFile
* @throws IOException 如果发生错误
*/
private File copyToTempFile(Uri uri, File tempFile) throws IOException {
    // 从 URI 获取输入流
    InputStream inputStream = getContentResolver().openInputStream(uri);

    if (inputStream == null) {
        throw new IOException("无法从 URI 获取输入流");
    }

    // 将流复制到临时文件
    FileUtils.copyInputStreamToFile(inputStream, tempFile);

    return tempFile;
}

```

第41.17节：通过 Intent 共享多个文件

传递给share()方法的字符串列表参数包含了你想要分享的所有文件的路径。

它基本上遍历这些路径，将它们添加到Uri中，并启动可以接受此类型文件的Activity。

```

public static void share(AppCompatActivity context, List<String> paths) {

    if (paths == null || paths.size() == 0) {
        return;
    }
    ArrayList<Uri> uris = new ArrayList<>();
    Intent intent = new Intent();
    intent.setAction(android.content.Intent.ACTION_SEND_MULTIPLE);
    intent.setType("*/*");
    for (String path : paths) {
        File file = new File(path);
        uris.add(Uri.fromFile(file));
    }
    intent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
    context.startActivity(intent);
}

```

第41.18节：使用Intent启动无绑定服务

服务是一个组件，它在后台（UI线程上）运行，不与用户直接交互。
未绑定的服务刚刚启动，且未绑定到任何活动的生命周期。

要启动服务，可以按下面的示例操作：

```

// 该Intent将用于启动服务
Intent i = new Intent(context, ServiceName.class);
// 可能向intent的extras中添加数据
i.putExtra("KEY1", "服务将使用的值");
context.startService(i);

```

你可以通过重写 onStartCommand() 来使用intent中的任何extras：

```

public class MyService extends Service {
    public MyService() {
    }
}

```

```

* @return the input tempFile for convenience
* @throws IOException if an error occurs
*/
private File copyToTempFile(Uri uri, File tempFile) throws IOException {
    // Obtain an input stream from the uri
    InputStream inputStream = getContentResolver().openInputStream(uri);

    if (inputStream == null) {
        throw new IOException("Unable to obtain input stream from URI");
    }

    // Copy the stream to the temp file
    FileUtils.copyInputStreamToFile(inputStream, tempFile);

    return tempFile;
}

```

Section 41.17: Sharing Multiple Files through Intent

The String List passed as a parameter to the share() method contains the paths of all the files you want to share.

It basically loops through the paths, adds them to Uri, and starts the Activity which can accept Files of this type.

```

public static void share(AppCompatActivity context, List<String> paths) {

    if (paths == null || paths.size() == 0) {
        return;
    }
    ArrayList<Uri> uris = new ArrayList<>();
    Intent intent = new Intent();
    intent.setAction(android.content.Intent.ACTION_SEND_MULTIPLE);
    intent.setType("*/*");
    for (String path : paths) {
        File file = new File(path);
        uris.add(Uri.fromFile(file));
    }
    intent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
    context.startActivity(intent);
}

```

Section 41.18: Start Unbound Service using an Intent

A Service is a component which runs in the background (on the UI thread) without direct interaction with the user.
An unbound Service is just started, and is not bound to the lifecycle of any Activity.

To start a Service you can do as shown in the example below:

```

// This Intent will be used to start the service
Intent i = new Intent(context, ServiceName.class);
// potentially add data to the intent extras
i.putExtra("KEY1", "Value to be used by the service");
context.startService(i);

```

You can use any extras from the intent by using an onStartCommand() override:

```

public class MyService extends Service {
    public MyService() {
    }
}

```

```

@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
    if (intent != null) {
        Bundle extras = intent.getExtras();
        String key1 = extras.getString("KEY1", "");
        if (key1.equals("服务将使用的值")) {
            //do something
        }
    }
    return START_STICKY;
}

@NoArgsConstructor
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

第41.19节：从Activity获取结果到Fragment

类似于从另一个Activity获取结果，你需要调用Fragment的startActivityForResult(Intent intent, int requestCode)方法。注意你不应该调用getActivity().startActivityForResult()，因为这会将结果返回给Fragment的父Activity。

接收结果可以使用Fragment的onActivityResult()方法。你需要确保Fragment的父Activity也重写了onActivityResult()并调用了其super实现。

在下面的示例中，ActivityOne包含FragmentOne，后者将启动ActivityTwo并期望从中获得结果。

ActivityOne

```

public class ActivityOne extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
    }

    // 您必须重写此方法，因为第二个活动总是会将其结果发送到此活动，然后再发送到片段
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
    }
}

```

activity_one.xml

```

<fragment android:name="com.example.FragmentOne"
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

FragmentOne

```

@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
    if (intent != null) {
        Bundle extras = intent.getExtras();
        String key1 = extras.getString("KEY1", "");
        if (key1.equals("Value to be used by the service")) {
            //do something
        }
    }
    return START_STICKY;
}

@NoArgsConstructor
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

Section 41.19: Getting a result from Activity to Fragment

Like Getting a result from another Activity you need to call the Fragment's method `startActivityForResult(Intent intent, int requestCode)`. note that you should not call `getActivity().startActivityForResult()` as this will take the result back to the Fragment's parent Activity.

Receiving the result can be done using the Fragment's method `onActivityResult()`. You need to make sure that the Fragment's parent Activity also overrides `onActivityResult()` and calls it's `super` implementation.

In the following example ActivityOne contains FragmentOne, which will start ActivityTwo and expect a result from it.

ActivityOne

```

public class ActivityOne extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);

        // You must override this method as the second Activity will always send its results to this
        // Activity and then to the Fragment
        @Override
        public void onActivityResult(int requestCode, int resultCode, Intent data) {
            super.onActivityResult(requestCode, resultCode, data);
        }
}

```

activity_one.xml

```

<fragment android:name="com.example.FragmentOne"
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

FragmentOne

```

public class FragmentOne extends Fragment {
    public static final int REQUEST_CODE = 11;
    public static final int RESULT_CODE = 12;
    public static final String EXTRA_KEY_TEST = "testKey";

    // 初始化并启动第二个Activity
    private void startSecondActivity() {
        Intent intent = new Intent(getActivity(), ActivityTwo.class);
        startActivityForResult(REQUEST_CODE, intent);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE && resultCode == RESULT_CODE) {
            String testResult = data.getStringExtra(EXTRA_KEY_TEST);
            // TODO: 处理你的额外数据
        }
    }
}

```

ActivityTwo

```

public class ActivityTwo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);

        private void closeActivity() {
            Intent intent = new Intent();
            intent.putExtra(FragmentOne.EXTRA_KEY_TEST, "Testing passing data back to ActivityOne");
            setResult(FragmentOne.RESULT_CODE, intent); // You can also send result without any data
            using setResult(int resultCode)
            finish();
        }
    }
}

```

```

public class FragmentOne extends Fragment {
    public static final int REQUEST_CODE = 11;
    public static final int RESULT_CODE = 12;
    public static final String EXTRA_KEY_TEST = "testKey";

    // Initializing and starting the second Activity
    private void startSecondActivity() {
        Intent intent = new Intent(getActivity(), ActivityTwo.class);
        startActivityForResult(REQUEST_CODE, intent);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE && resultCode == RESULT_CODE) {
            String testResult = data.getStringExtra(EXTRA_KEY_TEST);
            // TODO: Do something with your extra data
        }
    }
}

```

ActivityTwo

```

public class ActivityTwo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);

        private void closeActivity() {
            Intent intent = new Intent();
            intent.putExtra(FragmentOne.EXTRA_KEY_TEST, "Testing passing data back to ActivityOne");
            setResult(FragmentOne.RESULT_CODE, intent); // You can also send result without any data
            using setResult(int resultCode)
            finish();
        }
    }
}

```

第42章：碎片 (Fragments)

关于碎片及其相互通信机制的介绍

第42.1节：使用

Bundle从Activity向Fragment传递数据

所有碎片都应该有一个空构造函数（即无输入参数的构造方法）。因此，为了向正在创建的碎片传递数据，应该使用`setArguments()`方法。该方法接收一个`Bundle`，你将数据存储在其中，并将`Bundle`存储在参数中。随后，可以在碎片的`onCreate()`和`onCreateView()`回调中检索该`Bundle`。

活动：

```
Bundle bundle = new Bundle();
String myMessage = "Stack Overflow 很酷！";
bundle.putString("message", myMessage);
FragmentClass fragInfo = new FragmentClass();
fragInfo.setArguments(bundle);
transaction.replace(R.id.fragment_single, fragInfo);
transaction.commit();
```

Fragment：

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    String myValue = this.getArguments().getString("message");
    ...
}
```

第42.2节：newInstance() 模式

虽然可以创建带参数的片段构造函数，但Android在重新创建片段时（例如，当片段因Android自身原因被杀死后恢复）会内部调用无参构造函数。因此，不建议依赖带参数的构造函数。

为了确保您的预期片段参数始终存在，您可以使用静态的`newInstance()`方法来创建片段，并将您想要的参数放入一个`bundle`中，该`bundle`将在创建新实例时可用。

```
import android.os.Bundle;
import android.support.v4.app.Fragment;

public class 我的碎片 extends Fragment
{
    // 我们用于从参数中获取名称的标识符
    private static final String NAME_ARG = "name";

    private String mName;

    // 必需的
    public 我的碎片(){}

    // 静态构造方法。这是你应该实例化碎片的唯一方式
}
```

Chapter 42: Fragments

Introduction about Fragments and their intercommunication mechanism

Section 42.1: Pass data from Activity to Fragment using Bundle

All fragments should have an empty constructor (i.e. a constructor method having no input arguments). Therefore, in order to pass your data to the Fragment being created, you should use the `setArguments()` method. This method gets a bundle, which you store your data in, and stores the Bundle in the arguments. Subsequently, this Bundle can then be retrieved in `onCreate()` and `onCreateView()` call backs of the Fragment.

Activity:

```
Bundle bundle = new Bundle();
String myMessage = "Stack Overflow is cool!";
bundle.putString("message", myMessage);
FragmentClass fragInfo = new FragmentClass();
fragInfo.setArguments(bundle);
transaction.replace(R.id.fragment_single, fragInfo);
transaction.commit();
```

Fragment:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    String myValue = this.getArguments().getString("message");
    ...
}
```

Section 42.2: The newInstance() pattern

Although it is possible to create a fragment constructor with parameters, Android internally calls the zero-argument constructor when recreating fragments (for example, if they are being restored after being killed for Android's own reasons). For this reason, it is not advisable to rely on a constructor that has parameters.

To ensure that your expected fragment arguments are always present you can use a static `newInstance()` method to create the fragment, and put whatever parameters you want in to a bundle that will be available when creating a new instance.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;

public class MyFragment extends Fragment
{
    // Our identifier for obtaining the name from arguments
    private static final String NAME_ARG = "name";

    private String mName;

    // Required
    public MyFragment(){}

    // The static constructor. This is the only way that you should instantiate
    // the fragment yourself
}
```

```

public static 我的碎片 newInstance(final String name) {
    final 我的碎片 myFragment = new 我的碎片();
    // 下面的1是一个优化，表示将添加到此bundle中的参数数量。如果你知道要添加到bundle中的
    // 参数数量，// 它可以避免对底层映射的额外分配。如果不确定，可以构造一个不带参数的Bundle

    final Bundle args = new Bundle(1);

    // 这将参数作为参数存储在bundle中。注意，即使'name'参数为NULL，这也能正常工作，// 因此
    // 你应该在此时考虑该参数是否为必需，如果是，则检查是否为NULL并抛出适当的错误

    args.putString(NAME_ARG, name);

    myFragment.setArguments(args);
    return myFragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final Bundle arguments = getArguments();
    if (arguments == null || !arguments.containsKey(NAME_ARG)) {
        // 根据需要设置默认值或错误处理
    } 否则 {
        mName = arguments.getString(NAME_ARG);
    }
}

```

现在，在 Activity 中：

```

FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
MyFragment mFragment = MyFragment.newInstance("my name");
ft.replace(R.id.placeholder, mFragment);
//R.id.placeholder 是我们想要加载 fragment 的位置
ft.commit();

```

这种模式是确保在创建时将所有必要参数传递给 fragment 的最佳实践。

请注意，当系统销毁片段并稍后重新创建时，它会自动恢复其状态——但你必须为其提供一个onSaveInstanceState(Bundle)的实现。

第42.3节：使用返回栈和静态工厂模式在片段之间导航

首先，我们需要在开始时添加第一个Fragment，应该在我们的

ActivityOnCreate(方法实现)

```

if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
        .addToBackStack("fragmentA")
        .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
        .commit();
}

```

接下来，我们需要管理返回栈。最简单的方法是在我们的活动中添加一个函数，用于所有的FragmentTransactions。

```

public static MyFragment newInstance(final String name) {
    final MyFragment myFragment = new MyFragment();
    // The 1 below is an optimization, being the number of arguments that will
    // be added to this bundle. If you know the number of arguments you will add
    // to the bundle it stops additional allocations of the backing map. If
    // unsure, you can construct Bundle without any arguments
    final Bundle args = new Bundle(1);

    // This stores the argument as an argument in the bundle. Note that even if
    // the 'name' parameter is NULL then this will work, so you should consider
    // at this point if the parameter is mandatory and if so check for NULL and
    // throw an appropriate error if so
    args.putString(NAME_ARG, name);

    myFragment.setArguments(args);
    return myFragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final Bundle arguments = getArguments();
    if (arguments == null || !arguments.containsKey(NAME_ARG)) {
        // Set a default or error as you see fit
    } else {
        mName = arguments.getString(NAME_ARG);
    }
}

```

Now, in the Activity:

```

FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
MyFragment mFragment = MyFragment.newInstance("my name");
ft.replace(R.id.placeholder, mFragment);
//R.id.placeholder is where we want to load our fragment
ft.commit();

```

This pattern is a best practice to ensure that all the needed arguments will be passed to fragments on creation. Note that when the system destroys the fragment and re-creates it later, it will automatically restore its state - but you must provide it with an [onSaveInstanceState\(Bundle\)](#) implementation.

Section 42.3: Navigation between fragments using backstack and static fabric pattern

First of all, we need to add our first Fragment at the beginning, we should do it in the `onCreate()` method of our Activity:

```

if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
        .addToBackStack("fragmentA")
        .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
        .commit();
}

```

Next, we need to manage our backstack. The easiest way is using a function added in our activity that is used for all FragmentTransactions.

```

public void replaceFragment(Fragment fragment, String tag) {
    //获取当前放置在容器中的片段
    Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.container);

    //防止在顶部添加相同的片段
    if (currentFragment.getClass() == fragment.getClass()) {
        return;
    }

    //如果片段已经在栈中，我们可以弹出栈以防止栈无限增长
    if (getSupportFragmentManager().findFragmentByTag(tag) != null) {
        getSupportFragmentManager().popBackStack(tag, FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }

    //否则，只需替换碎片
    getSupportFragmentManager()
        .beginTransaction()
        .addToBackStack(tag)
        .replace(R.id.container, fragment, tag)
        .commit();
}

```

最后，我们应该重写 `onBackPressed()` 方法，以便在从后退栈中最后一个可用碎片返回时退出应用程序。

```

@Override
public void onBackPressed() {
    int fragmentsInStack = getSupportFragmentManager().getBackStackEntryCount();
    if (fragmentsInStack > 1) { // 如果有多个碎片，则弹出后退栈
        getSupportFragmentManager().popBackStack();
    } else if (fragmentsInStack == 1) { // 如果只剩一个碎片，则结束活动，防止留下空白屏幕
        finish();
    } 否则 {
        super.onBackPressed();
    }
}

```

在活动中的执行：

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

在活动外部执行（假设 `MainActivity` 是我们的活动）：

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

第42.4节：通过回调接口将事件发送回活动

如果需要从片段向活动发送事件，一种可能的解决方案是定义回调接口，并要求宿主活动实现该接口。

示例

当片段的按钮被点击时，向活动发回调

首先，定义回调接口：

```
public interface SampleCallback {
    void onButtonClicked();
}
```

```

public void replaceFragment(Fragment fragment, String tag) {
    //Get current fragment placed in container
    Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.container);

    //Prevent adding same fragment on top
    if (currentFragment.getClass() == fragment.getClass()) {
        return;
    }

    //If fragment is already on stack, we can pop back stack to prevent stack infinite growth
    if (getSupportFragmentManager().findFragmentByTag(tag) != null) {
        getSupportFragmentManager().popBackStack(tag, FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }

    //Otherwise, just replace fragment
    getSupportFragmentManager()
        .beginTransaction()
        .addToBackStack(tag)
        .replace(R.id.container, fragment, tag)
        .commit();
}

```

Finally, we should override `onBackPressed()` to exit the application when going back from the last Fragment available in the backstack.

```

@Override
public void onBackPressed() {
    int fragmentsInStack = getSupportFragmentManager().getBackStackEntryCount();
    if (fragmentsInStack > 1) { // If we have more than one fragment, pop back stack
        getSupportFragmentManager().popBackStack();
    } else if (fragmentsInStack == 1) { // Finish activity, if only one fragment left, to prevent
        leave empty screen
        finish();
    } else {
        super.onBackPressed();
    }
}

```

Execution in activity:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Execution outside activity (assuming `MainActivity` is our activity):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Section 42.4: Sending events back to an activity with callback interface

If you need to send events from fragment to activity, one of the possible solutions is to define callback interface and require that the host activity implement it.

Example

Send callback to an activity, when fragment's button clicked

First of all, define callback interface:

```
public interface SampleCallback {
    void onButtonClicked();
}
```

```
}
```

下一步是在片段中分配此回调：

```
public final class SampleFragment extends Fragment {

    private SampleCallback callback;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof SampleCallback) {
            callback = (SampleCallback) context;
        } else {
            throw new RuntimeException(context.toString()
                + " 必须实现 SampleCallback");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        callback = null;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        final View view = inflater.inflate(R.layout.sample, container, false);
        // 添加按钮的点击监听器
        view.findViewById(R.id.actionButton).setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                callback.onButtonClicked(); // 在这里调用回调
            }
        });
        return view;
    }
}
```

最后，在活动中实现回调：

```
public final class SampleActivity extends Activity implements SampleCallback {

    // ... 省略设置内容视图和展示片段的代码

    @Override
    public void onButtonClicked() {
        // 当片段的按钮被点击时调用
    }
}
```

第42.5节：为碎片之间的过渡添加动画

要为碎片之间的过渡添加动画，或为显示或隐藏碎片的过程添加动画，您可以使用FragmentManager来创建一个FragmentTransaction。

对于单个FragmentTransaction，有两种不同的方式来执行动画：您可以使用标准动画，也可以提供自定义动画。

```
}
```

Next step is to assign this callback in fragment:

```
public final class SampleFragment extends Fragment {

    private SampleCallback callback;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof SampleCallback) {
            callback = (SampleCallback) context;
        } else {
            throw new RuntimeException(context.toString()
                + " must implement SampleCallback");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        callback = null;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        final View view = inflater.inflate(R.layout.sample, container, false);
        // Add button's click listener
        view.findViewById(R.id.actionButton).setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                callback.onButtonClicked(); // Invoke callback here
            }
        });
        return view;
    }
}
```

And finally, implement callback in activity:

```
public final class SampleActivity extends Activity implements SampleCallback {

    // ... Skipped code with settings content view and presenting the fragment

    @Override
    public void onButtonClicked() {
        // Invoked when fragment's button has been clicked
    }
}
```

Section 42.5: Animate the transition between fragments

To animate the transition between fragments, or to animate the process of showing or hiding a fragment you use the FragmentManager to create a FragmentTransaction.

For a single FragmentTransaction, there are two different ways to perform animations: you can use a standard animation or you can supply your own custom animations.

标准动画通过调用FragmentTransaction.setTransition(int transit)来指定，并使用FragmentTransaction类中预定义的常量之一。在撰写本文时，这些常量包括：

```
FragmentTransaction.TRANSIT_NONE  
FragmentTransaction.TRANSIT_FRAGMENT_OPEN  
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE  
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

完整的事务可能如下所示：

```
getSupportFragmentManager()  
.beginTransaction()  
.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)  
.replace(R.id.contents, new MyFragment(), "MyFragmentTag")  
.commit();
```

自定义动画通过调用FragmentTransaction.setCustomAnimations(**int** enter, **int** exit)或FragmentTransaction.setCustomAnimations(**int** enter, **int** exit, **int** popEnter, **int** popExit)来指定。

对于不涉及从返回栈中弹出碎片的FragmentTransaction，将播放enter和exit动画。弹出返回栈中的碎片时，将播放pop Enter和popExit动画。

以下代码展示了如何通过滑出一个片段并滑入另一个片段来替换片段。

```
getSupportFragmentManager()  
.beginTransaction()  
.setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)  
.replace(R.id.contents, new MyFragment(), "MyFragmentTag")  
.commit();
```

XML 动画定义将使用objectAnimator标签。一个 slide_in_left.xml 的示例如下：

```
<?xml version="1.0" encoding="utf-8"?>  
<set>  
  <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"  
    android:propertyName="x"  
    android:valueType="floatType"  
    android:valueFrom="-1280"  
    android:valueTo="0"  
    android:duration="500"/>  
</set>
```

第42.6节：片段之间的通信

所有片段之间的通信必须通过一个活动（Activity）进行。片段**不能**直接相互通信，必须通过活动。

附加资源

- [如何实现 OnFragmentInteractionListener](#)
- [Android | 与其他片段通信](#)

在此示例中，我们有一个MainActivity，托管两个片段，SenderFragment和ReceiverFragment，分别用于

Standard animations are specified by calling FragmentTransaction.setTransition(**int** transit), and using one of the pre-defined constants available in the FragmentTransaction class. At the time of writing, these constants are:

```
FragmentTransaction.TRANSIT_NONE  
FragmentTransaction.TRANSIT_FRAGMENT_OPEN  
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE  
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

The complete transaction might look something like this:

```
getSupportFragmentManager()  
.beginTransaction()  
.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)  
.replace(R.id.contents, new MyFragment(), "MyFragmentTag")  
.commit();
```

Custom animations are specified by calling either FragmentTransaction.setCustomAnimations(**int** enter, **int** exit) or FragmentTransaction.setCustomAnimations(**int** enter, **int** exit, **int** popEnter, **int** popExit).

The enter and exit animations will be played for FragmentTransactions that do not involve popping fragments off of the back stack. The popEnter and popExit animations will be played when popping a fragment off of the back stack.

The following code shows how you would replace a fragment by sliding out one fragment and sliding the other one in its place.

```
getSupportFragmentManager()  
.beginTransaction()  
.setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)  
.replace(R.id.contents, new MyFragment(), "MyFragmentTag")  
.commit();
```

The XML animation definitions would use the objectAnimator tag. An example of **slide_in_left.xml** might look something like this:

```
<?xml version="1.0" encoding="utf-8"?>  
<set>  
  <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"  
    android:propertyName="x"  
    android:valueType="floatType"  
    android:valueFrom="-1280"  
    android:valueTo="0"  
    android:duration="500"/>  
</set>
```

Section 42.6: Communication between Fragments

All communications between Fragments must go via an Activity. Fragments **CANNOT** communicate with each other without an Activity.

Additional Resources

- [How to implement OnFragmentInteractionListener](#)
- [Android | Communicating With Other Fragments](#)

In this sample, we have a MainActivity that hosts two fragments, SenderFragment and ReceiverFragment, for

发送和接收消息（此处为简单的字符串）。

SenderFragment中的一个按钮启动发送消息的过程。当ReceiverFragment接收到消息时，其中文本视图（TextView）会被更新。

以下是带有注释的 MainActivity 代码片段，解释了重要的代码行：

```
// 我们的 MainActivity 实现了 SenderFragment 定义的接口。这使得  
// 片段可以与活动进行通信  
public class MainActivity extends AppCompatActivity implements SenderFragment.SendMessageListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /**  
     * 当我们点击 SenderFragment 中的按钮时调用此方法  
     * @param message 由 SenderFragment 发送的消息  
     */  
    @Override  
    public void onSendMessage(String message) {  
        // 使用 SupportFragmentManager 和 fragment 的 id 查找我们的 ReceiverFragment  
        ReceiverFragment receiverFragment = (ReceiverFragment)  
            getSupportFragmentManager().findFragmentById(R.id.fragment_receiver);  
  
        // 确保该 fragment 存在  
        if (receiverFragment != null) {  
            // 通过调用 ReceiverFragment 的公共方法将此消息发送给它  
            receiverFragment.showMessage(message);  
        }  
    }  
}
```

MainActivity 在一个 LinearLayout 中托管两个 fragment：

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/activity_main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="com.naru.fragmentcommunication.MainActivity">  
  
<fragment  
    android:id="@+id/fragment_sender"  
    android:name="com.naru.fragmentcommunication.SenderFragment"  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_weight="1"  
    tools:layout="@layout/fragment_sender" />  
  
<fragment  
    android:id="@+id/fragment_receiver"  
    android:name="com.naru.fragmentcommunication.ReceiverFragment"
```

sending and receiving a message (a simple String in this case) respectively.

A Button in SenderFragment initiates the process of sending the message. A TextView in the ReceiverFragment is updated when the message is received by it.

Following is the snippet for the MainActivity with comments explaining the important lines of code:

```
// Our MainActivity implements the interface defined by the SenderFragment. This enables  
// communication from the fragment to the activity  
public class MainActivity extends AppCompatActivity implements SenderFragment.SendMessageListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /**  
     * This method is called when we click on the button in the SenderFragment  
     * @param message The message sent by the SenderFragment  
     */  
    @Override  
    public void onSendMessage(String message) {  
        // Find our ReceiverFragment using the SupportFragmentManager and the fragment's id  
        ReceiverFragment receiverFragment = (ReceiverFragment)  
            getSupportFragmentManager().findFragmentById(R.id.fragment_receiver);  
  
        // Make sure that such a fragment exists  
        if (receiverFragment != null) {  
            // Send this message to the ReceiverFragment by calling its public method  
            receiverFragment.showMessage(message);  
        }  
    }  
}
```

The layout file for the MainActivity hosts two fragments inside a LinearLayout :

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/activity_main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="com.naru.fragmentcommunication.MainActivity">  
  
<fragment  
    android:id="@+id/fragment_sender"  
    android:name="com.naru.fragmentcommunication.SenderFragment"  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_weight="1"  
    tools:layout="@layout/fragment_sender" />  
  
<fragment  
    android:id="@+id/fragment_receiver"  
    android:name="com.naru.fragmentcommunication.ReceiverFragment"
```

```

        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_receiver" />
    </LinearLayout>

```

SenderFragment暴露了一个接口SendMessageListener，帮助MainActivity知道SenderFragment中的按钮何时被点击。

以下是SenderFragment的代码片段，解释了重要的代码行：

```

public class SenderFragment extends Fragment {

    private SendMessageListener commander;

    /**
     * 创建此接口是为了在活动和碎片之间进行通信。任何实现此接口的活动 * 都能够接收该碎片发送的消息。
     */

    public interface SendMessageListener {
        void onSendMessage(String message);
    }

    /**
     * API 级别 >= 23
     * <p>
     * 当片段附加到活动时调用此方法。这里的方法将帮助我们初始化接口的引用变量“commander”
     * 'SendMessageListener'
     *
     * @param context
     */
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        // 尝试将 context 强制转换为我们的接口 SendMessageListener，即检查该活动是否实现了 SendMessageListener。如果没有，则抛出 ClassCastException。
        try {
            commander = (SendMessageListener) context;
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                + "必须实现 SendMessageListener 接口");
        }
    }

    /**
     * API 级别 < 23
     * <p>
     * 当片段附加到活动时调用此方法。这里的方法将帮助我们初始化接口的引用变量“commander”
     * 'SendMessageListener'
     *
     * @param activity
     */
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        // 尝试将 context 强制转换为我们的接口 SendMessageListener，即检查该活动是否实现了 SendMessageListener。如果没有，则抛出 ClassCastException。
        try {

```

```

        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_receiver" />
    </LinearLayout>

```

The SenderFragment exposes an interface SendMessageListener that helps the MainActivity know when Button in the SenderFragment was clicked.

Following is the code snippet for the SenderFragment explaining the important lines of code:

```

public class SenderFragment extends Fragment {

    private SendMessageListener commander;

    /**
     * This interface is created to communicate between the activity and the fragment. Any activity
     * which implements this interface will be able to receive the message that is sent by this
     * fragment.
     */
    public interface SendMessageListener {
        void onSendMessage(String message);
    }

    /**
     * API LEVEL >= 23
     * <p>
     * This method is called when the fragment is attached to the activity. This method here will
     * help us to initialize our reference variable, 'commander' , for our interface
     * 'SendMessageListener'
     *
     * @param context
     */
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        // Try to cast the context to our interface SendMessageListener i.e. check whether the
        // activity implements the SendMessageListener. If not a ClassCastException is thrown.
        try {
            commander = (SendMessageListener) context;
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                + "must implement the SendMessageListener interface");
        }
    }

    /**
     * API LEVEL < 23
     * <p>
     * This method is called when the fragment is attached to the activity. This method here will
     * help us to initialize our reference variable, 'commander' , for our interface
     * 'SendMessageListener'
     *
     * @param activity
     */
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        // Try to cast the context to our interface SendMessageListener i.e. check whether the
        // activity implements the SendMessageListener. If not a ClassCastException is thrown.
        try {

```

```

commander = (SendMessageListener) activity;
} catch (ClassCastException e) {
    throw new ClassCastException(activity.toString()
        + "必须实现 SendMessageListener 接口");
}
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    // 为发送者片段加载视图。
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // 初始化按钮并设置点击监听器
    Button send = (Button) view.findViewById(R.id.bSend);
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            // 检查我们是否成功初始化了接口引用
            if (commander != null) {

                // 调用我们的接口方法。这使我们能够调用活动中实现的方法,
                // 从那里我们可以将消息发送到 ReceiverFragment。
                commander.onSendMessage("来自发送者片段的问候！");
            }
        }
    });
}

return view;
}
}

```

发送者片段的布局文件：

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <Button
        android:id="@+id/bSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="发送"
        android:layout_gravity="center_horizontal" />
</LinearLayout>

```

接收者片段很简单，公开了一个简单的公共方法来更新其TextView。当主活动（MainActivity）接收到来自发送者片段（SenderFragment）的消息时，它会调用接收者片段的这个公共方法。以下是接收者片段的代码片段，并附有解释重要代码行的注释：

```

public class ReceiverFragment extends Fragment {
    TextView tvMessage;
    @Nullable

```

```

    commander = (SendMessageListener) activity;
} catch (ClassCastException e) {
    throw new ClassCastException(activity.toString()
        + "must implement the SendMessageListener interface");
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize button and a click listener on it
    Button send = (Button) view.findViewById(R.id.bSend);
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            // Sanity check whether we were able to properly initialize our interface reference
            if (commander != null) {

                // Call our interface method. This enables us to call the implemented method
                // in the activity, from where we can send the message to the ReceiverFragment.
                commander.onSendMessage("HELLO FROM SENDER FRAGMENT!");
            }
        }
    });
}

return view;
}
}

```

The layout file for the SenderFragment:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <Button
        android:id="@+id/bSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SEND"
        android:layout_gravity="center_horizontal" />
</LinearLayout>

```

The ReceiverFragment is simple and exposes a simple public method to update its TextView. When the MainActivity receives the message from the SenderFragment it calls this public method of the ReceiverFragment

Following is the code snippet for the ReceiverFragment with comments explaining the important lines of code :

```

public class ReceiverFragment extends Fragment {
    TextView tvMessage;
    @Nullable

```

```

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    // 为发送者片段加载视图。
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // 初始化TextView
    tvMessage = (TextView) view.findViewById(R.id.tvReceivedMessage);

    return view;
}

/**
 * 该方法由 MainActivity 调用，当它从 SenderFragment 接收到消息时触发。
 * 该方法帮助将 TextView 中的文本更新为 SenderFragment 发送的消息。
 * @param message 由 SenderFragment 通过 MainActivity 发送的消息。
 */
public void showMessage(String message) {
    tvMessage.setText(message);
}

```

ReceiverFragment :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
    <TextView
        android:id="@+id/tvReceivedMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="等待消息！" />
</LinearLayout>

```

```

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize the TextView
    tvMessage = (TextView) view.findViewById(R.id.tvReceivedMessage);

    return view;
}

/**
 * Method that is called by the MainActivity when it receives a message from the SenderFragment.
 * This method helps update the text in the TextView to the message sent by the SenderFragment.
 * @param message Message sent by the SenderFragment via the MainActivity.
 */
public void showMessage(String message) {
    tvMessage.setText(message);
}

```

The layout file for the ReceiverFragment :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
    <TextView
        android:id="@+id/tvReceivedMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Waiting for message!" />
</LinearLayout>

```

第43章：按钮

第43.1节：在XML中为一个或多个视图使用相同的点击事件

当我们在布局中创建任何视图时，可以使用 `android:onClick` 属性引用关联的活动或片段中的方法来处理点击事件。

XML布局

```
<Button android:id="@+id/button"
...
// onClick 应该引用你在活动或片段中的方法
    android:onClick="doSomething" />

// 注意，这适用于任何继承自 View 的类，而不仅仅是 Button
<ImageView android:id="@+id/image"
...
    android:onClick="doSomething" />
```

活动/片段代码

在你的代码中，创建你命名的方法，其中 `v` 将是被触摸的视图，并对每个调用此方法的视图执行操作。

```
public void doSomething(View v) {
    switch(v.getId()) {
        case R.id.button:
            // 按钮被点击，执行某些操作。
            break;
        case R.id.image:
            // 图片被点击，执行其他操作。
            break;
    }
}
```

如果你愿意，也可以为每个视图使用不同的方法（在这种情况下，当然不必检查 ID）。

第43.2节：定义外部监听器

我什么时候应该使用它

- 当内联监听器中的代码过多，导致你的方法/类变得丑陋且难以阅读
- 你想在应用的多个元素（视图）中执行相同的操作

为此，你需要创建一个实现了View API中某个监听器的类。

例如，长按任何元素时显示帮助：

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
```

Chapter 43: Button

Section 43.1: Using the same click event for one or more Views in the XML

When we create any View in layout, we can use the `android:onClick` attribute to reference a method in the associated activity or fragment to handle the click events.

XML Layout

```
<Button android:id="@+id/button"
...
// onClick should reference the method in your activity or fragment
    android:onClick="doSomething" />

// Note that this works with any class which is a subclass of View, not just Button
<ImageView android:id="@+id/image"
...
    android:onClick="doSomething" />
```

Activity/fragment code

In your **code**, create the method you named, where `v` will be the view that was touched, and do something for each view that calls this method.

```
public void doSomething(View v) {
    switch(v.getId()) {
        case R.id.button:
            // Button was clicked, do something.
            break;
        case R.id.image:
            // Image was clicked, do something else.
            break;
    }
}
```

If you want, you can also use different method for each View (in this case, of course, you don't have to check for the ID).

Section 43.2: Defining external Listener

When should I use it

- When the code inside an inline listener is too big and your method / class becomes ugly and hard to read
- You want to perform same action in various elements (view) of your app

To achieve this you need to create a class implementing one of the listeners in the [View API](#).

For example, give help when long click on any element:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
```

```
public void onLongClick(View v) {  
    // 显示帮助提示或弹出窗口  
}  
}
```

然后你只需要在你的Activity中拥有一个属性或变量来使用它：

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);  
  
button1.setOnClickListener(helpListener);  
button2.setOnClickListener(helpListener);  
label.setOnClickListener(helpListener);  
button1.setOnClickListener(helpListener);
```

注意：在单独的类中定义监听器有一个缺点，就是不能直接访问类字段，因此除非将属性设为公共或定义getter，否则需要通过构造函数传递数据（上下文、视图）。

第43.3节：内联onClickListener

假设我们有一个按钮（可以通过编程创建，或者使用findViewById()等方法从视图绑定）

```
Button btnOK = (...)
```

现在，创建一个匿名类并内联设置它。

```
btnOK.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // 在这里执行操作...  
    }  
});
```

第43.4节：自定义按钮样式

有多种方式可以自定义按钮的外观。此示例展示了几种选项：

选项0：使用ThemeOverlay（目前最简单/最快捷的方法）

在你的样式文件中创建一个新样式：

styles.xml

```
<resources>  
    <style name="mybutton" parent="ThemeOverlay.AppCompat.Light">  
        <!-- 自定义禁用颜色的colorButtonNormal -->  
        <item name="colorButtonNormal">@color/colorbuttonnormal</item>  
        <!-- 自定义启用颜色的colorAccent -->  
        <item name="colorButtonNormal">@color/coloraccent</item>  
    </style>  
</resources>
```

然后在放置按钮的布局中（例如MainActivity）：

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
public void onLongClick(View v) {  
    // show help toast or popup  
}
```

Then you just need to have an attribute or variable in your Activity to use it:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);  
  
button1.setOnClickListener(helpListener);  
button2.setOnClickListener(helpListener);  
label.setOnClickListener(helpListener);  
button1.setOnClickListener(helpListener);
```

NOTE: defining listeners in separated class has one disadvantage, it cannot access class fields directly, so you need to pass data (context, view) through constructor unless you make attributes public or define getters.

Section 43.3: inline onClickListener

Say we have a button (we can create it programmatically, or bind it from a view using findViewById(), etc...)

```
Button btnOK = (...)
```

Now, create an anonymous class and set it inline.

```
btnOK.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // Do stuff here...  
    }  
});
```

Section 43.4: Customizing Button style

There are many possible ways of customizing the look of a Button. This example presents several options:

Option 0: Use ThemeOverlay (currently the easiest/quickest way)

Create a new style in your styles file:

styles.xml

```
<resources>  
    <style name="mybutton" parent="ThemeOverlay.AppCompat.Light">  
        <!-- customize colorButtonNormal for the disable color -->  
        <item name="colorButtonNormal">@color/colorbuttonnormal</item>  
        <!-- customize colorAccent for the enabled color -->  
        <item name="colorButtonNormal">@color/coloraccent</item>  
    </style>  
</resources>
```

Then in the layout where you place your button (e.g. MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_gravity="center_horizontal"
android:gravity="center_horizontal"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<Button
    android:id="@+id/mybutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello"
    android:theme="@style/mybutton"
    style="@style/Widget.AppCompat.Button.Colored"/>

</LinearLayout>

```

选项1：创建您自己的按钮样式

在 values/styles.xml 中，为您的按钮创建一个新样式：

styles.xml

```

<resources>
    <style name="mybuttonstyle" parent="@android:style/Widget.Button">
        <item name="android:gravity">center_vertical|center_horizontal</item>
        <item name="android:textColor">#FFFFFF</item>
        <item name="android:shadowColor">#FF000000</item>
        <item name="android:shadowDx">0</item>
        <item name="android:shadowDy">-1</item>
        <item name="android:shadowRadius">0.2</item>
        <item name="android:textSize">16dip</item>
        <item name="android:textStyle">bold</item>
        <item name="android:background">@drawable/button</item>
    </style>
</resources>

```

然后在放置按钮的布局中（例如在 MainActivity 中）：

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

```

```

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_gravity="center_horizontal"
android:gravity="center_horizontal"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<Button
    android:id="@+id/mybutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello"
    android:theme="@style/mybutton"
    style="@style/Widget.AppCompat.Button.Colored"/>

</LinearLayout>

```

Option 1: Create your own button style

In values/styles.xml, create a new style for your button:

styles.xml

```

<resources>
    <style name="mybuttonstyle" parent="@android:style/Widget.Button">
        <item name="android:gravity">center_vertical|center_horizontal</item>
        <item name="android:textColor">#FFFFFF</item>
        <item name="android:shadowColor">#FF000000</item>
        <item name="android:shadowDx">0</item>
        <item name="android:shadowDy">-1</item>
        <item name="android:shadowRadius">0.2</item>
        <item name="android:textSize">16dip</item>
        <item name="android:textStyle">bold</item>
        <item name="android:background">@drawable/button</item>
    </style>
</resources>

```

Then in the layout where you place your button (e.g. in MainActivity):

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

```

```
<Button  
    android:id="@+id/mybutton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello"  
    android:theme="@style/mybuttonstyle"/>  
  
</LinearLayout>
```

选项2：为按钮的每个状态分配一个drawable

在drawable文件夹中创建一个名为'mybuttondrawable.xml'的xml文件，以定义按钮每个状态的drawable资源：

drawable/mybutton.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
    <item  
        android:state_enabled="false"  
        android:drawable="@drawable/mybutton_disabled" />  
    <item  
        android:state_pressed="true"  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_pressed" />  
    <item  
        android:state_focused="true"  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_focused" />  
    <item  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_enabled" />  
</selector>
```

这些drawable可以是图片（例如mybutton_disabled.png）或由你定义并存储在drawable文件夹中的xml文件。例如：

drawable/mybutton_disabled.xml

```
<?xml version="1.0" encoding="utf-8"?>  
  
    <shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">  
        <gradient  
            android:startColor="#F2F2F2"  
            android:centerColor="#A4A4A4"  
            android:endColor="#F2F2F2"  
            android:angle="90" />  
        <padding android:left="7dp"  
            android:top="7dp"  
            android:right="7dp"  
            android:bottom="7dp" />  
        <stroke  
            android:width="2dp"  
            android:color="#FFFFFF" />  
        <corners android:radius="8dp" />  
    </shape>
```

然后在放置按钮的布局中（例如MainActivity）：

activity_main.xml

```
<Button  
    android:id="@+id/mybutton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello"  
    android:theme="@style/mybuttonstyle"/>  
  
</LinearLayout>
```

Option 2: Assign a drawable for each of your button states

Create an xml file into drawable folder called 'mybuttondrawable.xml' to define the drawable resource of each of your button states:

drawable/mybutton.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
    <item  
        android:state_enabled="false"  
        android:drawable="@drawable/mybutton_disabled" />  
    <item  
        android:state_pressed="true"  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_pressed" />  
    <item  
        android:state_focused="true"  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_focused" />  
    <item  
        android:state_enabled="true"  
        android:drawable="@drawable/mybutton_enabled" />  
</selector>
```

Each of those drawables may be images (e.g. mybutton_disabled.png) or xml files defined by you and stored in the drawables folder. For instance:

drawable/mybutton_disabled.xml

```
<?xml version="1.0" encoding="utf-8"?>  
  
    <shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">  
        <gradient  
            android:startColor="#F2F2F2"  
            android:centerColor="#A4A4A4"  
            android:endColor="#F2F2F2"  
            android:angle="90" />  
        <padding android:left="7dp"  
            android:top="7dp"  
            android:right="7dp"  
            android:bottom="7dp" />  
        <stroke  
            android:width="2dp"  
            android:color="#FFFFFF" />  
        <corners android:radius="8dp" />  
    </shape>
```

Then in the layout where you place your button (e.g. MainActivity):

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/mybutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:background="@drawable/mybuttondrawable"/>

</LinearLayout>

```

选项3：将您的按钮样式添加到应用主题中

您可以在应用主题的定义中（位于 values/styles.xml）覆盖默认的 Android 按钮样式。

styles.xml

```

<resources>
    <style name="AppTheme" parent="android:Theme">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:button">@style/mybutton</item>
    </style>

    <style name="mybutton" parent="android:style/Widget.Button">
        <item name="android:gravity">center_vertical|center_horizontal</item>
        <item name="android:textColor">#FFFFFF</item>
        <item name="android:shadowColor">#FF000000</item>
        <item name="android:shadowDx">0</item>
        <item name="android:shadowDy">-1</item>
        <item name="android:shadowRadius">0.2</item>
        <item name="android:textSize">16dip</item>
        <item name="android:textStyle">bold</item>
        <item name="android:background">@drawable/anydrawable</item>
    </style>
</resources>

```

选项4：通过编程方式在默认按钮样式上叠加颜色

只需在您的活动中找到按钮并应用颜色滤镜：

```

Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)

```

您可以查看不同的混合模式那里以及很好的示例那里。

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/mybutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:background="@drawable/mybuttondrawable"/>

</LinearLayout>

```

Option 3: Add your button style to your App theme

You can override the default android button style in the definition of your app theme (in values/styles.xml).

styles.xml

```

<resources>
    <style name="AppTheme" parent="android:Theme">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:button">@style/mybutton</item>
    </style>

    <style name="mybutton" parent="android:style/Widget.Button">
        <item name="android:gravity">center_vertical|center_horizontal</item>
        <item name="android:textColor">#FFFFFF</item>
        <item name="android:shadowColor">#FF000000</item>
        <item name="android:shadowDx">0</item>
        <item name="android:shadowDy">-1</item>
        <item name="android:shadowRadius">0.2</item>
        <item name="android:textSize">16dip</item>
        <item name="android:textStyle">bold</item>
        <item name="android:background">@drawable/anydrawable</item>
    </style>
</resources>

```

Option 4: Overlay a color on the default button style programmatically

Just find you button in your activity and apply a color filter:

```

Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)

```

You can check different blending modes [here](#) and nice examples [here](#).

第43.5节：自定义点击监听器以防止多次快速点击

为了防止按钮在短时间内触发多次（比如1秒内点击2次，如果流程不受控可能会导致严重问题），可以实现一个自定义的 `SingleClickListener`。

该ClickListener设置了一个特定的时间间隔作为阈值（例如1000毫秒）。

当按钮被点击时，会检查触发事件是否在你定义的时间间隔内执行过，如果没有则触发该事件。

```
public class SingleClickListener implements View.OnClickListener {  
  
    protected int defaultInterval;  
    private long lastTimeClicked = 0;  
  
    public SingleClickListener() {  
        this(1000);  
    }  
  
    public SingleClickListener(int minInterval) {  
        this.defaultInterval = minInterval;  
    }  
  
    @Override  
    public void onClick(View v) {  
        if (SystemClock.elapsedRealtime() - lastTimeClicked < defaultInterval) {  
            return;  
        }  
        lastTimeClicked = SystemClock.elapsedRealtime();  
        performClick(v);  
    }  
  
    public abstract void performClick(View v);  
}
```

在类中，`SingleClickListener` 与相关的按钮关联

```
myButton = (Button) findViewById(R.id.my_button);  
myButton.setOnClickListener(new SingleClickListener() {  
    @Override  
    public void performClick(View view) {  
        // 执行操作  
    }  
});
```

第43.6节：使用布局定义点击动作

当我们在布局中创建按钮时，可以使用 `android:onClick` 属性引用代码中的方法来处理点击事件。

Button

```
<Button  
    android:width="120dp"  
    android:height="wrap_content"  
    android:text="点击我"
```

Section 43.5: Custom Click Listener to prevent multiple fast clicks

In order to **prevent a button from firing multiple times within a short period of time** (let's say 2 clicks within 1 second, which may cause serious problems if the flow is not controlled), one can implement a custom `SingleClickListener`.

This ClickListener sets a specific time interval as threshold (for instance, 1000ms).

When the button is clicked, a check will be ran to see if the trigger was executed in the past amount of time you defined, and if not it will trigger it.

```
public class SingleClickListener implements View.OnClickListener {  
  
    protected int defaultInterval;  
    private long lastTimeClicked = 0;  
  
    public SingleClickListener() {  
        this(1000);  
    }  
  
    public SingleClickListener(int minInterval) {  
        this.defaultInterval = minInterval;  
    }  
  
    @Override  
    public void onClick(View v) {  
        if (SystemClock.elapsedRealtime() - lastTimeClicked < defaultInterval) {  
            return;  
        }  
        lastTimeClicked = SystemClock.elapsedRealtime();  
        performClick(v);  
    }  
  
    public abstract void performClick(View v);  
}
```

And in the class, the `SingleClickListener` is associated to the Button at stake

```
myButton = (Button) findViewById(R.id.my_button);  
myButton.setOnClickListener(new SingleClickListener() {  
    @Override  
    public void performClick(View view) {  
        // do stuff  
    }  
});
```

Section 43.6: Using the layout to define a click action

When we create a button in layout, we can use the `android:onClick` attribute to reference a method in code to handle clicks.

Button

```
<Button  
    android:width="120dp"  
    android:height="wrap_content"  
    android:text="Click me"
```

```
        android:onClick="handleClick" />
```

然后在你的活动中，创建 handleClick 方法。

```
public void handleClick(View v) {  
    // 执行相应操作。  
}
```

第43.7节：监听长按事件

要捕捉长按事件并使用它，您需要为按钮提供合适的监听器：

```
View.OnLongClickListener listener = new View.OnLongClickListener() {  
    public boolean onLongClick(View v) {  
        Button clickedButton = (Button) v;  
        String buttonText = clickedButton.getText().toString();  
        Log.v(TAG, "button long pressed --> " + buttonText);  
        return true;  
    }  
};  
  
button.setOnLongClickListener(listener);
```

```
        android:onClick="handleClick" />
```

Then in your activity, create the handleClick method.

```
public void handleClick(View v) {  
    // Do whatever.  
}
```

Section 43.7: Listening to the long click events

To catch a long click and use it you need to provide appropriate listener to button:

```
View.OnLongClickListener listener = new View.OnLongClickListener() {  
    public boolean onLongClick(View v) {  
        Button clickedButton = (Button) v;  
        String buttonText = clickedButton.getText().toString();  
        Log.v(TAG, "button long pressed --> " + buttonText);  
        return true;  
    }  
};  
  
button.setOnLongClickListener(listener);
```

第44章：模拟器

第44.1节：截屏

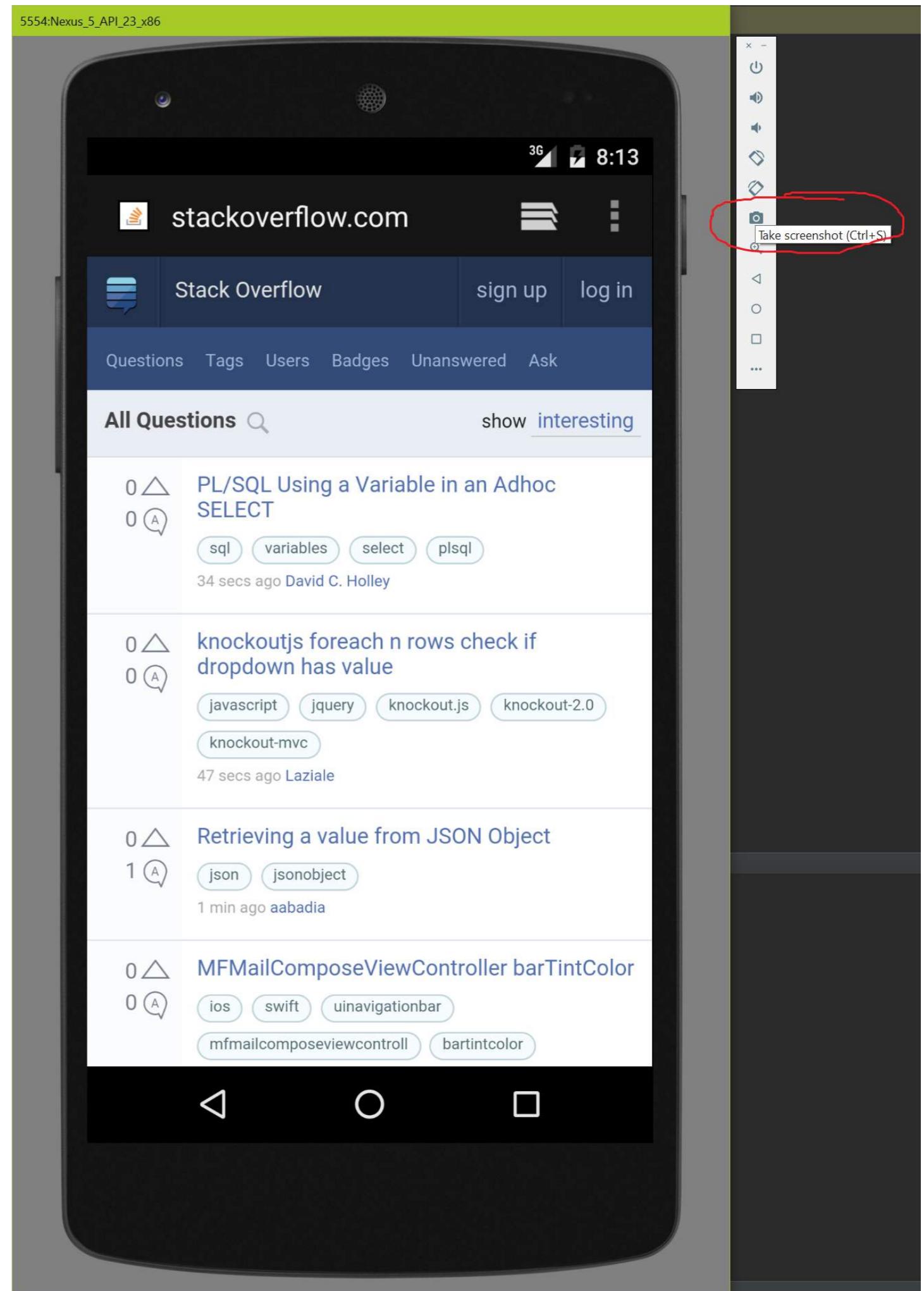
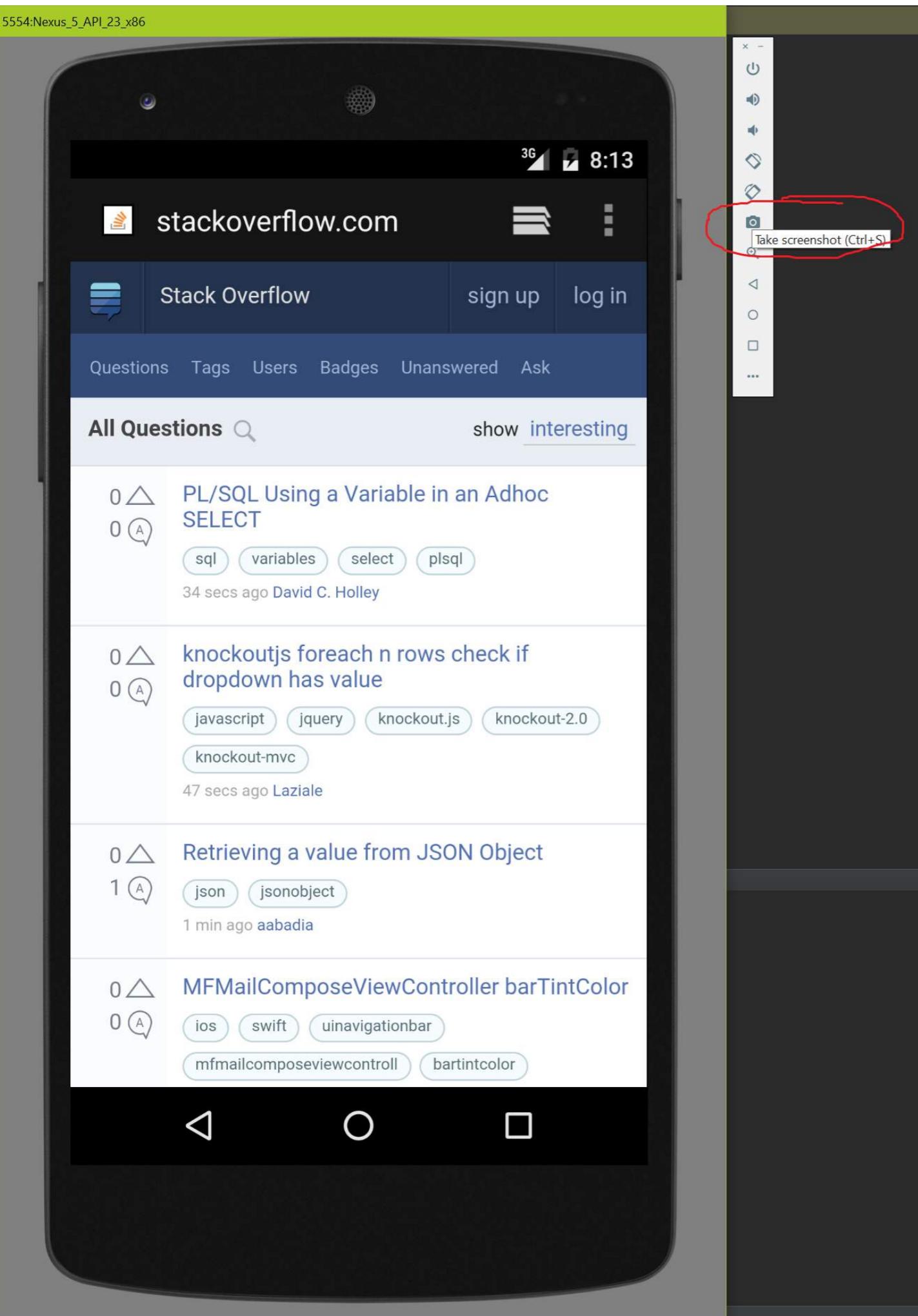
如果您想从Android模拟器（2.0）截取屏幕截图，只需按下
点击侧边栏上的相机图标：

Ctrl + **S** 或者你

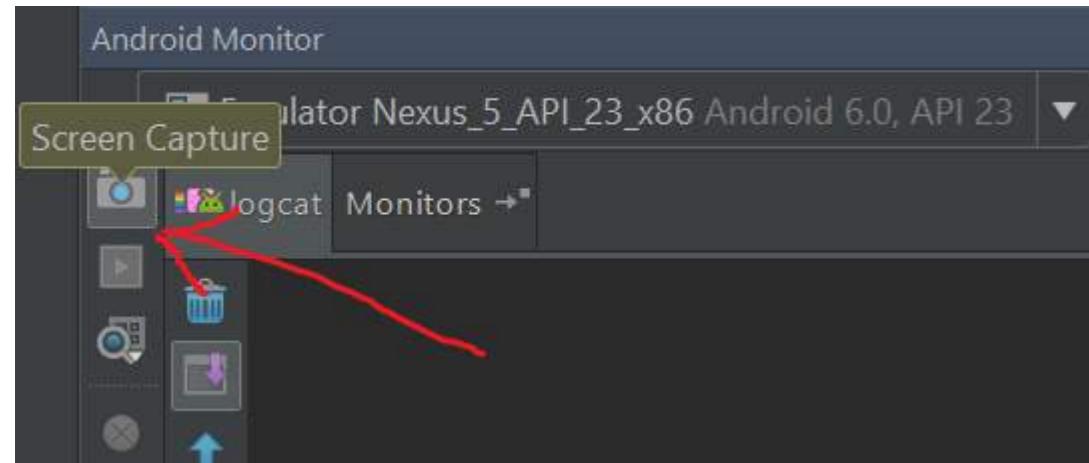
Chapter 44: Emulator

Section 44.1: Taking screenshots

If you want to take a screenshot from the Android Emulator (2.0), then you just need to press **Ctrl** + **S** or you click on the camera icon on the side bar:



如果你使用的是较旧版本的安卓模拟器，或者你想从真实设备截屏，那么你需要点击安卓监视器中的相机图标：

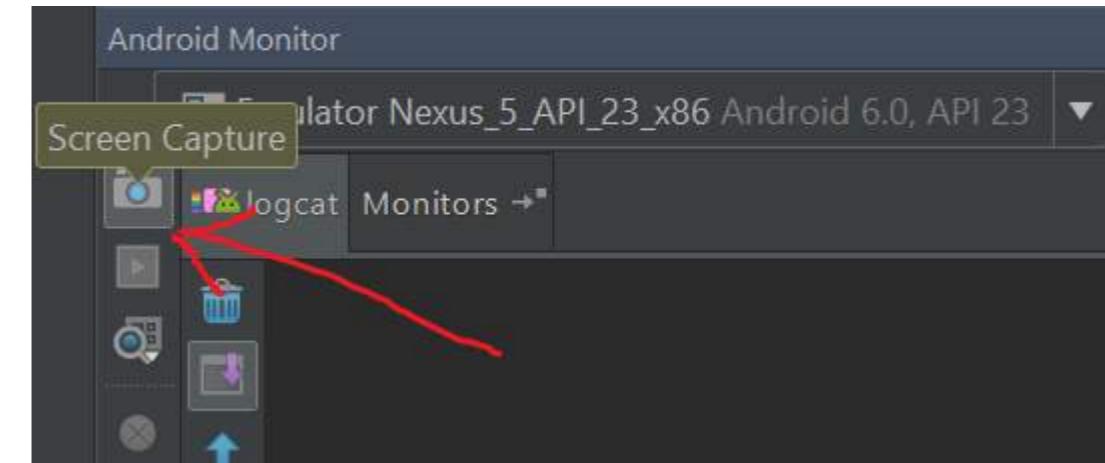


请再次确认你选择了正确的设备，因为这是一个常见的陷阱。

截屏后，你可以选择为其添加以下装饰（也请参见下图）：

1. 截屏周围的设备边框。
2. 设备边框下方的投影阴影。
3. 设备边框和截屏上的屏幕眩光。

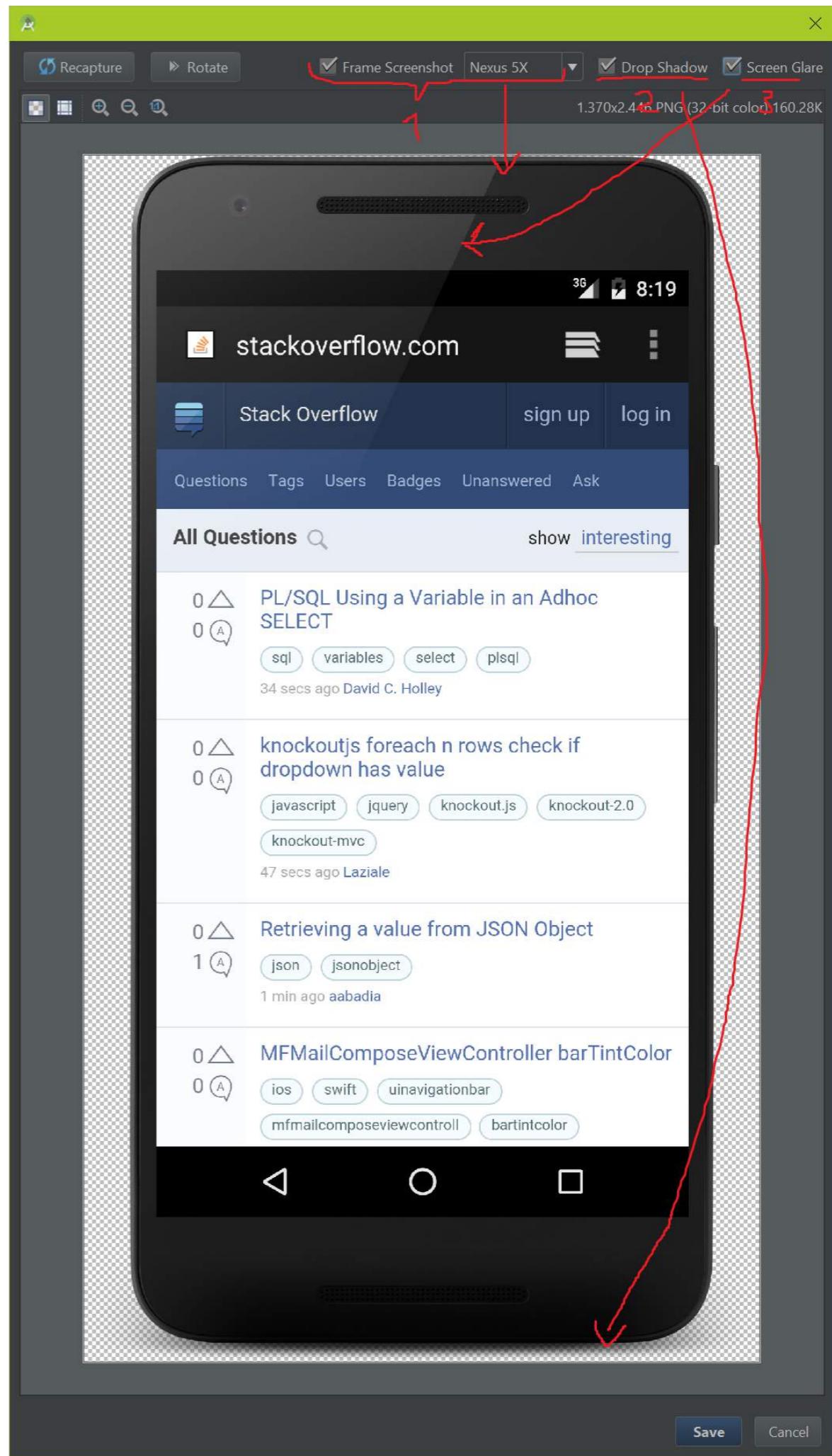
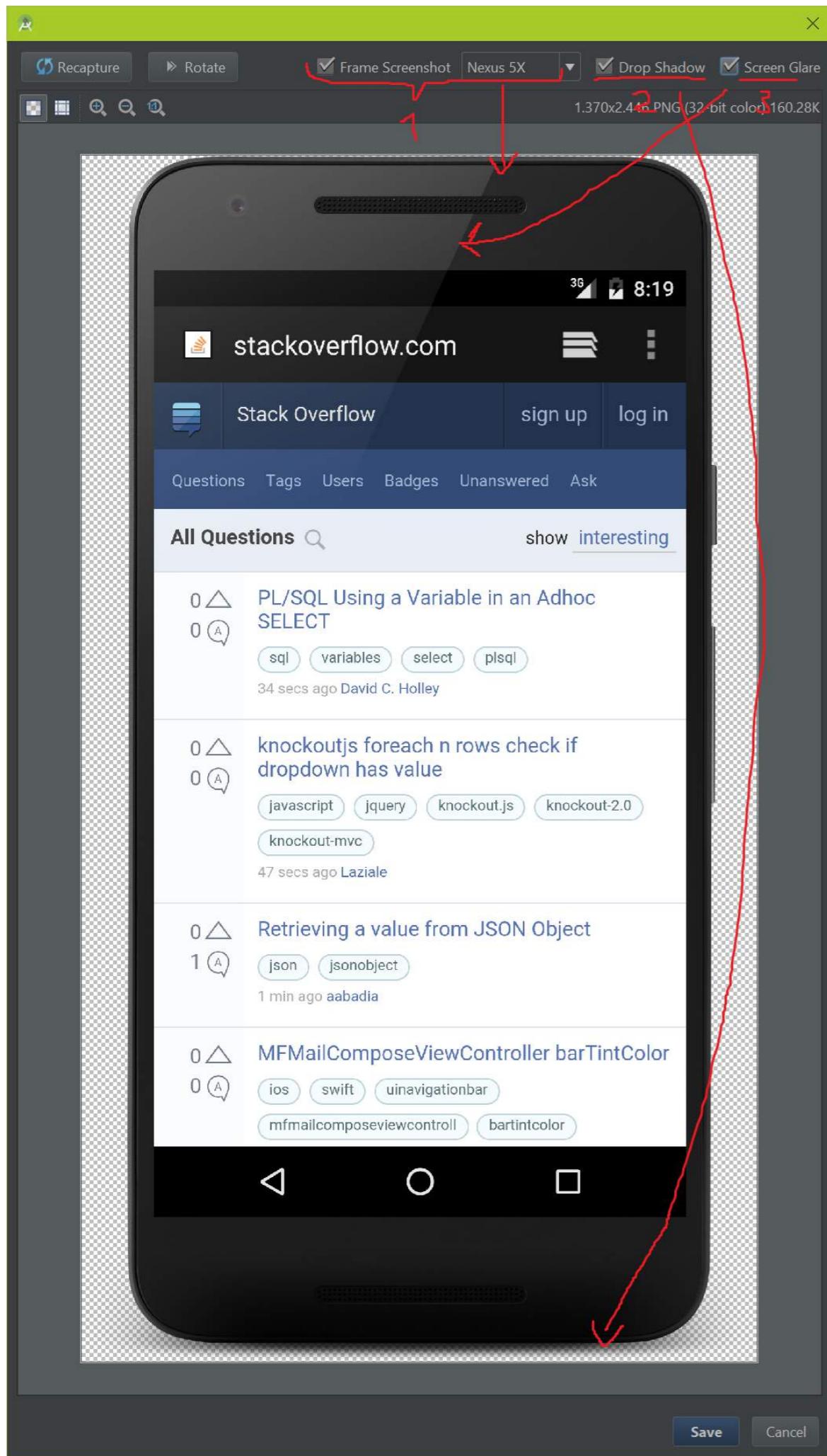
If you use an older version of the Android Emulator or you want to take a screenshot from a real device, then you need to click on the camera icon in the Android Monitor:



Double check that you have selected the right device, because this is a common pitfall.

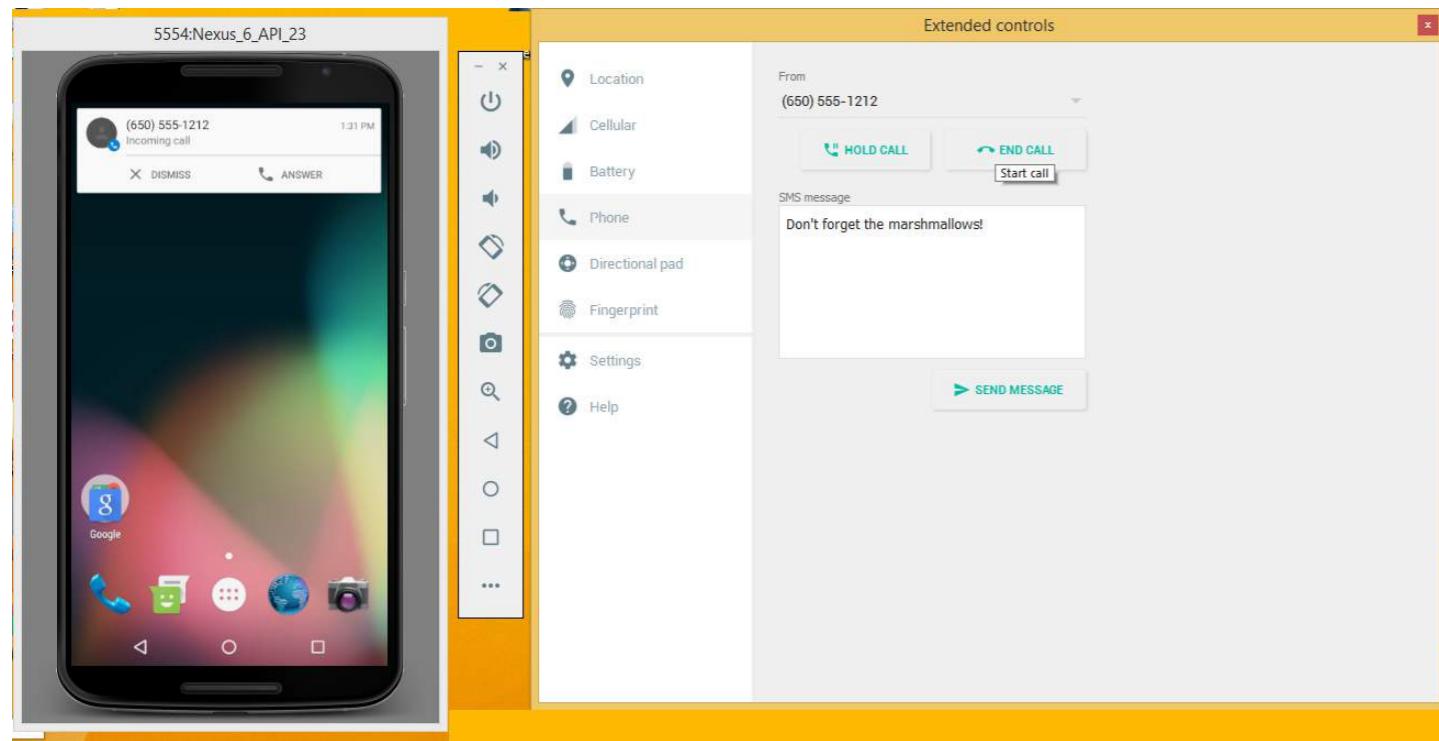
After taking a screenshot, you can optionally add the following decorations to it (also see the image below):

1. A device frame around the screenshot.
2. A drop shadow below the device frame.
3. A screen glare across device frame and screenshot.



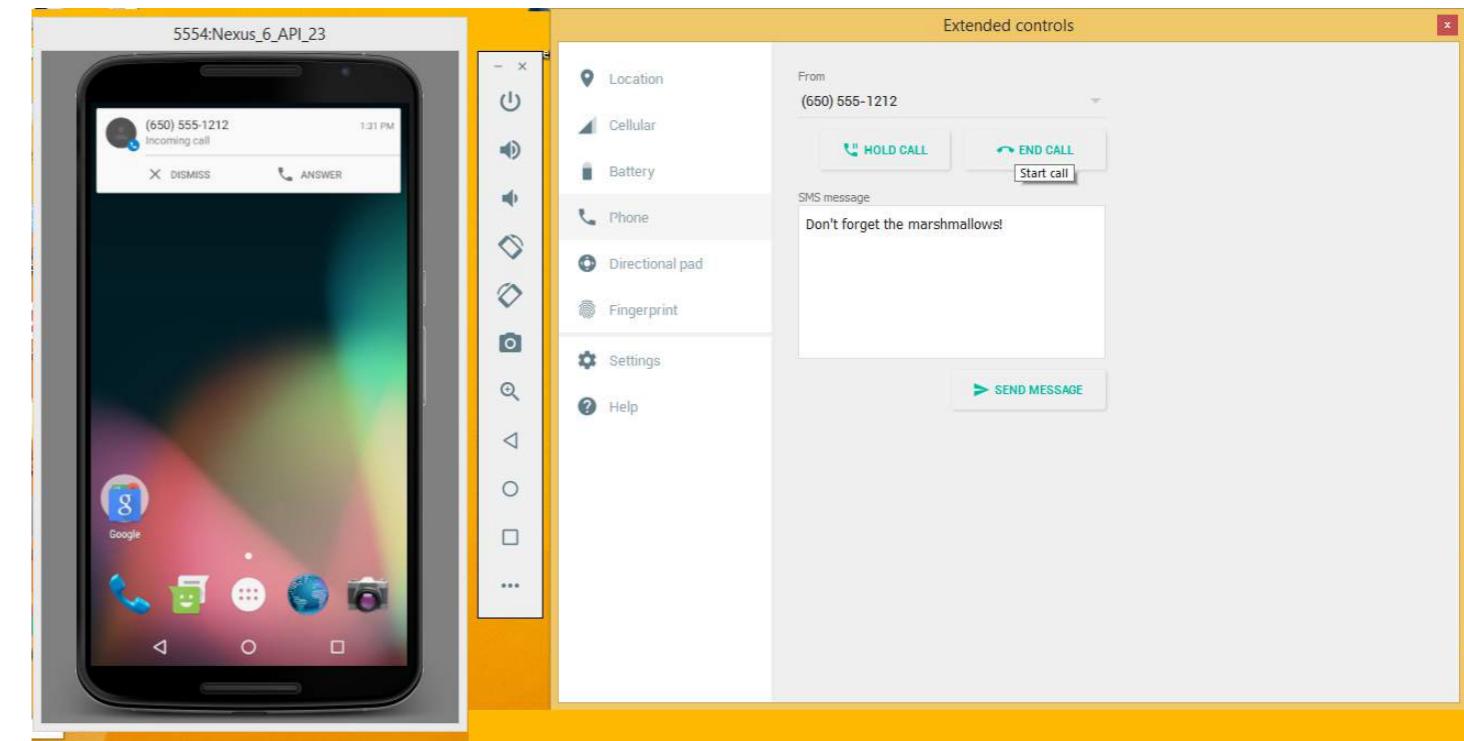
第44.2节：模拟通话

要模拟电话通话，按下由三个点表示的“扩展控制”按钮，选择“电话”，然后选择“通话”。您也可以选择更改电话号码。



Section 44.2: Simulate call

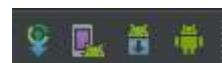
To simulate a phone call, press the 'Extended controls' button indicated by three dots, choose 'Phone' and select 'Call'. You can also optionally change the phone number.



第44.3节：打开AVD管理器

安装SDK后，您可以通过命令行使用`android avd`打开AVD管理器。

您也可以通过Android Studio访问AVD管理器，路径为工具(Tools) > Android > AVD管理器(AVD Manager)，或者点击工具栏中的AVD管理器图标，该图标在下图中为第二个。



第44.4节：启动模拟器时解决错误

首先，确保您已在BIOS设置中启用了“虚拟化(Virtualization)”。

启动Android SDK管理器(SDK Manager)选择附加组件(Extras) 然后选择Intel硬件加速执行管理器(Intel Hardware Accelerated Execution Manager)并等待下载完成。如果仍然无法使用，打开您的SDK文件夹并运行
`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe`。

按照屏幕上的指示完成安装。

或者对于 OS X，你可以这样在没有屏幕提示的情况下操作：

`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

如果你的 CPU 不支持 VT-x 或 SVM，则无法使用基于 x86 的 Android 镜像。请改用基于 ARM 的镜像。

安装完成后，通过打开命令提示符窗口并运行以下命令来确认虚拟化驱动程序是否正常运行：`sc query intelhaxm`

Section 44.3: Open the AVD Manager

Once the SDK installed, you can open the AVD Manager from the command line using `android avd`.

You can also access AVD Manager from Android studio using Tools > Android > AVD Manager or by clicking on the AVD Manager icon in the toolbar which is the second in the screenshot below.



Section 44.4: Resolving Errors while starting emulator

First of all, ensure that you've enabled the '**Virtualization**' in your BIOS setup.

Start the **Android SDK Manager**, select **Extras** and then select **Intel Hardware Accelerated Execution Manager** and wait until your download completes. If it still doesn't work, open your SDK folder and run
`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe`.

Follow the on-screen instructions to complete installation.

Or for OS X you can do it without onscreen prompts like this:

`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

If your CPU does not support VT-x or SVM, you can not use x86-based Android images. Please use ARM-based images instead.

After installation completed, confirm that the virtualization driver is operating correctly by opening a command prompt window and running the following command: `sc query intelhaxm`

要运行带有虚拟机加速的基于 x86 的模拟器：如果你从命令行运行模拟器，只需指定一个基于 x86 的 AVD：emulator -avd <avd_name>

如果你正确执行了上述所有步骤，那么你应该能够正常看到带有 HAXM 的 AVD 启动。

To run an x86-based emulator with VM acceleration: If you are running the emulator from the command line, just specify an x86-based AVD: emulator -avd <avd_name>

If you follow all the steps mentioned above correctly, then surely you should be able to see your AVD with HAXM coming up normally.

第45章：服务

服务在后台运行，以执行长时间运行的操作或为远程进程工作。服务不提供任何用户界面，它仅在后台运行且不接受用户输入。例如，服务可以在用户使用其他应用时在后台播放音乐，或者它可以从互联网下载数据而不会阻塞用户与 Android 设备的交互。

第45.1节：服务的生命周期

服务生命周期具有以下回调

- `onCreate()` :

当服务首次创建时执行，用于设置您可能需要的初始配置。此方法仅在服务尚未运行时执行。

- `onStartCommand()` :

每次由其他组件（如 Activity 或 BroadcastReceiver）调用 `startService()` 时执行。

使用此方法时，服务将一直运行，直到您调用 `stopSelf()` 或 `stopService()`。请注意，无论您调用 `onStartCommand()` 多少次，`stopSelf()` 和 `stopService()` 方法都必须只调用一次才能停止服务。

- `onBind()` :

当组件调用 `bindService()` 时执行，并返回一个 `IBinder` 实例，提供与服务的通信通道。只要有客户端绑定，调用 `bindService()` 将保持服务运行。

- `onDestroy()` :

当服务不再使用时执行，允许释放已分配的资源。

需要注意的是，在服务的生命周期中，可能会调用其他回调方法，例如 `onConfigurationChanged()` 和 `onLowMemory()`

<https://developer.android.com/guide/components/services.html>

Chapter 45: Service

A Service runs in **background** to perform long-running operations or to perform work for remote processes. A service does not provide any user interface it runs only in background with User's input. For example a service can play music in the background while the user is in a different App, or it might download data from the internet without blocking user's interaction with the Android device.

Section 45.1: Lifecycle of a Service

The services lifecycle has the following callbacks

- `onCreate()` :

Executed when the service is first created in order to set up the initial configurations you might need. This method is executed only if the service is not already running.

- `onStartCommand()` :

Executed every time `startService()` is invoked by another component, like an Activity or a BroadcastReceiver. When you use this method, the Service will run until you call `stopSelf()` or `stopService()`. Note that regardless of how many times you call `onStartCommand()`, the methods `stopSelf()` and `stopService()` must be invoked only once in order to stop the service.

- `onBind()` :

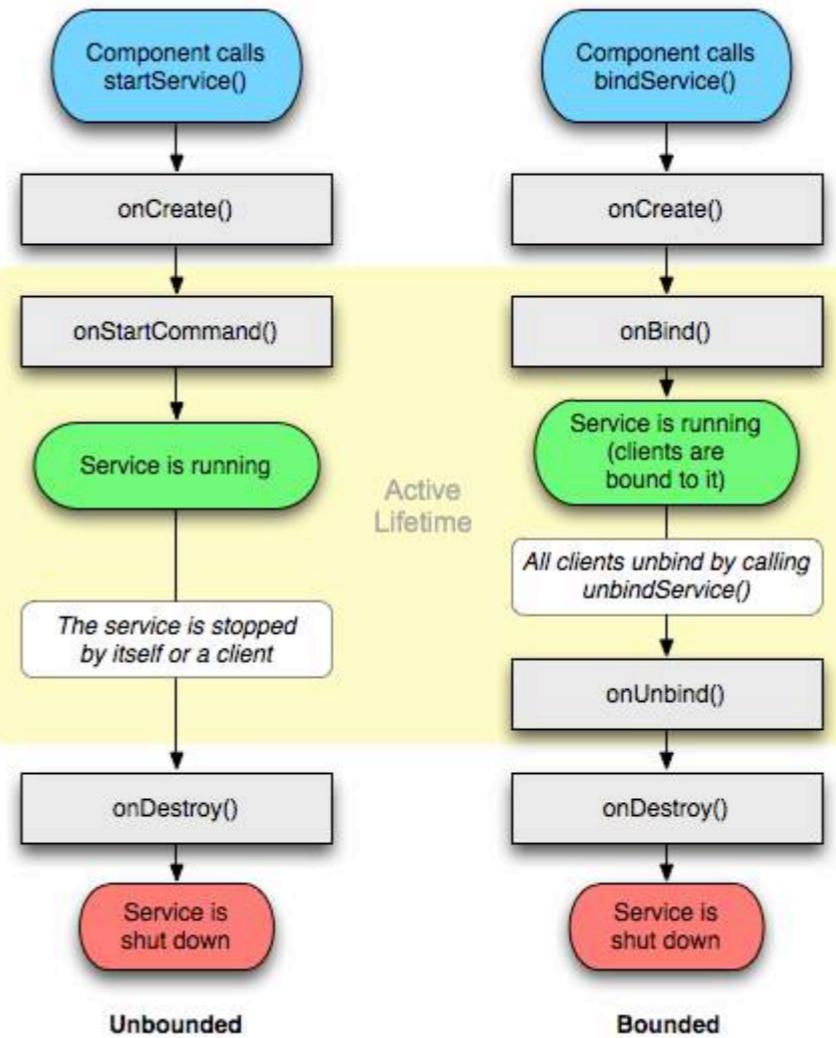
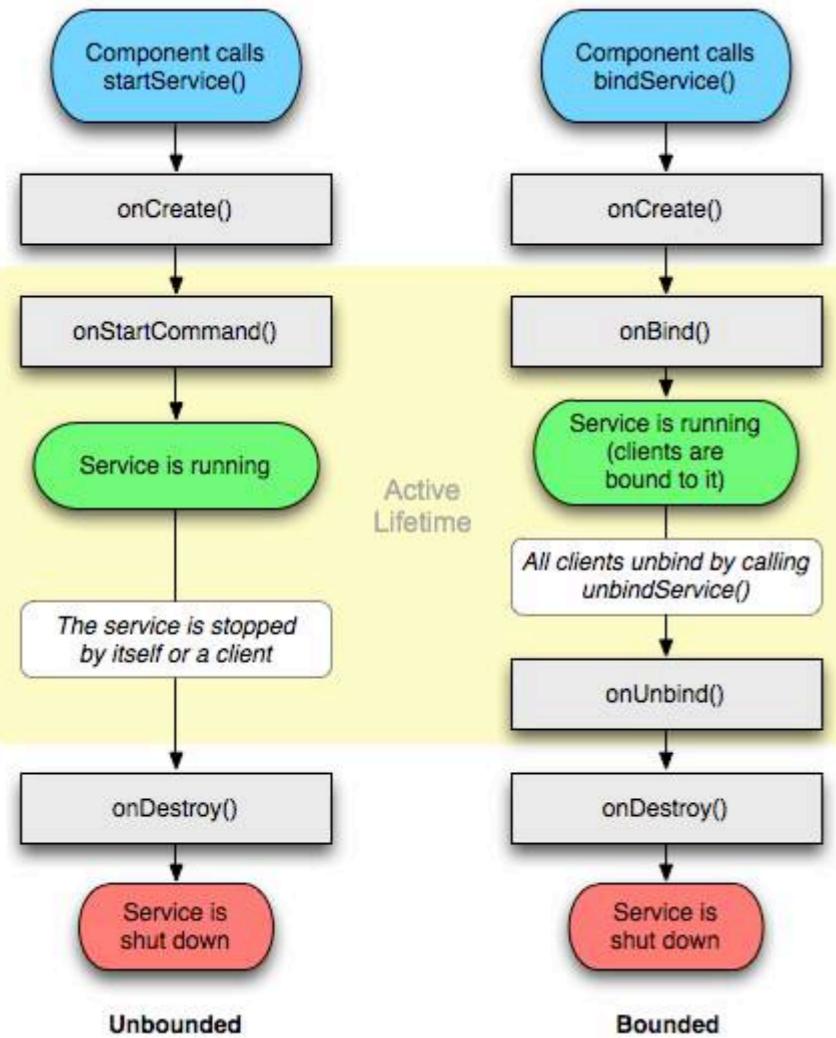
Executed when a component calls `bindService()` and returns an instance of `IBinder`, providing a communication channel to the Service. A call to `bindService()` will keep the service running as long as there are clients bound to it.

- `onDestroy()` :

Executed when the service is no longer in use and allows for disposal of resources that have been allocated.

It is important to note that during the lifecycle of a service other callbacks might be invoked such as `onConfigurationChanged()` and `onLowMemory()`

<https://developer.android.com/guide/components/services.html>



第45.2节：定义服务的进程

android:process 字段定义了服务运行的进程名称。通常，应用程序的所有组件都运行在为该应用程序创建的默认进程中。然而，组件可以通过其自己的 process 属性覆盖默认设置，从而允许你将应用程序分布在多个进程中。

如果分配给该属性的名称以冒号 (':') 开头，服务将运行在其自己的独立进程中。

```
<service
    android:name="com.example.appName"
    android:process=":externalProcess" />
```

如果进程名称以小写字母开头，服务将运行在该名称的全局进程中，前提是它有权限这样做。这允许不同应用程序中的组件共享一个进程，从而减少资源使用。

第45.3节：创建无绑定服务

首先要做的是将服务添加到AndroidManifest.xml中，放在<application>标签内：

```
<application ...>
    ...
    <service
        android:name=".RecordingService" />
```

Section 45.2: Defining the process of a service

The android:process field defines the name of the process where the service is to run. Normally, all components of an application run in the default process created for the application. However, a component can override the default with its own process attribute, allowing you to spread your application across multiple processes.

If the name assigned to this attribute begins with a colon (:), the service will run in its own separate process.

```
<service
    android:name="com.example.appName"
    android:process=":externalProcess" />
```

If the process name begins with a lowercase character, the service will run in a global process of that name, provided that it has permission to do so. This allows components in different applications to share a process, reducing resource usage.

Section 45.3: Creating an unbound service

The first thing to do is to add the service to AndroidManifest.xml, inside the <application> tag:

```
<application ...>
    ...
    <service
        android:name=".RecordingService" />
```

```

<!--"enabled" 标签指定系统是否可以实例化该服务 --
"true" -->
    <!--如果可以，则为"true"；如果不可以，则为"false"。默认值为"true"。-->
    android:enabled="true"
        <!--exported 标签指定其他应用程序的组件是否可以调用 --
->
        <!--服务或与其交互—如果可以则为"true"，如果不可以则为"false"。当值-
->
        <!--如果是"false"，则仅限于同一应用程序的组件或具有相同用户的应用程序
-->
    <!--ID 可以启动服务或绑定到它。-->
    android:exported="false" />

```

如果您打算将服务类管理在单独的包中（例如：.AllServices.RecordingService），那么您需要指定服务所在的位置。
因此，在上述情况下，我们将修改：

```
android:name=".RecordingService"
```

到

```
android:name=".AllServices.RecordingService"
```

或者最简单的方法是指定完整的包名。

然后我们创建实际的服务类：

```

public class RecordingService extends Service {
    private int NOTIFICATION = 1; // 我们通知的唯一标识符

    public static boolean isRunning = false;
    public static RecordingService instance = null;

    private NotificationManager notificationManager = null;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate(){
instance = this;
isRunning = true;

notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        // 如果用户选择此通知，启动我们的活动的 PendingIntent
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0, new Intent(this,
MainActivity.class), 0);

        // 设置在通知面板中显示的视图信息。
    }
}

```

```

<!--"enabled" tag specifies Whether or not the service can be instantiated by the system --
"true" -->
    <!--if it can be, and "false" if not. The default value is "true".-->
    android:enabled="true"
        <!--exported tag specifies Whether or not components of other applications can invoke the --
->
        <!--service or interact with it — "true" if they can, and "false" if not. When the value-
->
        <!--is "false"，only components of the same application or applications with the same user
-->
    <!--ID can start the service or bind to it.-->
    android:exported="false" />

</application>

```

If you intend to manage your service class in a separate package (eg: .AllServices.RecordingService) then you will need to specify where your service is located. So, in above case we will modify:

```
android:name=".RecordingService"
```

to

```
android:name=".AllServices.RecordingService"
```

or the easiest way of doing so is to specify the full package name.

Then we create the actual service class:

```

public class RecordingService extends Service {
    private int NOTIFICATION = 1; // Unique identifier for our notification

    public static boolean isRunning = false;
    public static RecordingService instance = null;

    private NotificationManager notificationManager = null;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate(){
        instance = this;
        isRunning = true;

        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        // The PendingIntent to launch our activity if the user selects this notification
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0, new Intent(this,
MainActivity.class), 0);

        // Set the info for the views that show in the notification panel.
    }
}

```

```

Notification notification = new NotificationCompat.Builder(this)
    .setSmallIcon(R.mipmap.ic_launcher) // 状态图标
    .setTicker("服务运行中...") // 时间戳
    .setContentTitle("我的应用") // 条目标签
    .setContentText("服务运行中...") // 条目内容
    .setContentIntent(contentIntent) // 条目被点击时发送的意图
    .setOngoing(true) // 设置为持续状态 (禁用滑动删除)
    .build();

    // 以前台模式启动服务
startForeground(NOTIFICATION, notification);

    return START_STICKY;
}

@Override
public void onDestroy(){
isRunning = false;
instance = null;

notificationManager.cancel(NOTIFICATION); // 移除通知
super.onDestroy();
}

public void doSomething(){
Toast.makeText(getApplicationContext(), "服务正在执行操作...", Toast.LENGTH_SHORT).show();
}
}

```

该服务所做的全部工作是在运行时显示通知，并且当调用其 `doSomething()` 方法时可以显示吐司提示。

如你所见，它被实现为一个 `singleton`，跟踪自身的实例——但没有通常的静态单例工厂方法，因为服务本质上是单例的，并且由意图创建。该实例对外部来说很有用，可以在服务运行时获取对服务的“句柄”。

最后，我们需要从一个活动中启动和停止该服务：

```

public void startOrStopService(){
    if( RecordingService.isRunning ){
        // 停止服务
        Intent intent = new Intent(this, RecordingService.class);
        stopService(intent);
    }
    else {
        // 启动服务
        Intent intent = new Intent(this, RecordingService.class);
        startService(intent);
    }
}

```

在此示例中，服务的启动和停止由同一方法控制，取决于其当前状态。

```

Notification notification = new NotificationCompat.Builder(this)
    .setSmallIcon(R.mipmap.ic_launcher) // the status icon
    .setTicker("Service running...") // the status text
    .setWhen(System.currentTimeMillis()) // the time stamp
    .setContentTitle("My App") // the label of the entry
    .setContentText("Service running...") // the content of the entry
    .setContentIntent(contentIntent) // the intent to send when the entry is clicked
    .setOngoing(true) // make persistent (disable swipe-away)
    .build();

    // Start service in foreground mode
startForeground(NOTIFICATION, notification);

    return START_STICKY;
}

@Override
public void onDestroy(){
isRunning = false;
instance = null;

notificationManager.cancel(NOTIFICATION); // Remove notification
super.onDestroy();
}

public void doSomething(){
Toast.makeText(getApplicationContext(), "Doing stuff from service...", Toast.LENGTH_SHORT).show();
}
}

```

All this service does is show a notification when it's running, and it can display toasts when its `doSomething()` method is called.

As you'll notice, it's implemented as a `singleton`, keeping track of its own instance - but without the usual static singleton factory method because services are naturally singletons and are created by intents. The instance is useful to the outside to get a "handle" to the service when it's running.

Last, we need to start and stop the service from an activity:

```

public void startOrStopService(){
    if( RecordingService.isRunning ){
        // Stop service
        Intent intent = new Intent(this, RecordingService.class);
        stopService(intent);
    }
    else {
        // Start service
        Intent intent = new Intent(this, RecordingService.class);
        startService(intent);
    }
}

```

In this example, the service is started and stopped by the same method, depending on its current state.

我们也可以从活动中调用 doSomething() 方法：

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

第45.4节：启动服务

启动服务非常简单，只需在活动中调用带有意图的 startService 即可：

```
Intent intent = new Intent(this, MyService.class); //将 MyService 替换为你的服务名称
intent.putExtra(Intent.EXTRA_TEXT, "Some text"); //添加任何要传递给服务的额外数据
startService(intent); //调用 startService 启动服务。
```

第45.5节：借助 Binder 创建绑定服务

创建一个继承Service类的类，并在重写的方法onBind中返回你的本地绑定器实例：

```
public class LocalService extends Service {
    // 给予客户端的绑定器
    private final IBinder mBinder = new LocalBinder();

    /**
     * 用于客户端绑定器的类。因为我们知道该服务总是在与其客户端相同的进程中运行，
     * 所以不需要处理IPC。
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // 返回此LocalService实例，以便客户端可以调用公共方法
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

然后在你的活动中，在onStart回调中绑定服务，使用ServiceConnection实例，并在onStop中解绑：

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
    }
}
```

We can also invoke the doSomething() method from our activity:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Section 45.4: Starting a Service

Starting a service is very easy, just call startService with an intent, from within an Activity:

```
Intent intent = new Intent(this, MyService.class); //substitute MyService with the name of your
service
intent.putExtra(Intent.EXTRA_TEXT, "Some text"); //add any extra data to pass to the service
startService(intent); //Call startService to start the service.
```

Section 45.5: Creating Bound Service with help of Binder

Create a class which extends Service class and in overridden method onBind return your local binder instance:

```
public class LocalService extends Service {
    // Binder given to clients
    private final IBinder mBinder = new LocalBinder();

    /**
     * Class used for the client Binder. Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // Return this instance of LocalService so clients can call public methods
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

Then in your activity bind to service in onStart callback, using ServiceConnection instance and unbind from it in onStop:

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
    }
}
```

```

// 绑定到本地服务
Intent intent = new Intent(this, LocalService.class);
bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
}

@Override
protected void onStop() {
    super.onStop();
    // 解绑服务
    if (mBound) {
        unbindService(mConnection);
        mBound = false;
    }
}

/** 定义服务绑定的回调，传递给 bindService() */
private ServiceConnection mConnection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName className,
                                   IBinder service) {
        // 我们已绑定到 LocalService，将 IBinder 强制转换并获取 LocalService 实例
        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName arg0) {
        mBound = false;
    }
};

```

```

// Bind to LocalService
Intent intent = new Intent(this, LocalService.class);
bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
}

@Override
protected void onStop() {
    super.onStop();
    // Unbind from the service
    if (mBound) {
        unbindService(mConnection);
        mBound = false;
    }
}

/** Defines callbacks for service binding, passed to bindService() */
private ServiceConnection mConnection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName className,
                                   IBinder service) {
        // We've bound to LocalService, cast the IBinder and get LocalService instance
        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName arg0) {
        mBound = false;
    }
};

```

第45.6节：创建远程服务（通过AIDL）

通过.aidl文件描述您的服务访问接口：

```

// IRemoteService.aidl
package com.example.android;

// 在此处使用import语句声明任何非默认类型

/** 示例服务接口 */
interface IRemoteService {
    /** 请求此服务的进程ID，以便进行一些操作。 */
    int getPid();
}

```

现在构建应用程序后，SDK工具将生成相应的.java文件。该文件将包含实现我们aidl接口的Stub类，我们需要继承它：

```

public class RemoteService extends Service {

    private final IRemoteService.Stub binder = new IRemoteService.Stub() {
        @Override
        public int getPid() throws RemoteException {
            return Process.myPid();
        }
    };
}

```

Section 45.6: Creating Remote Service (via AIDL)

Describe your service access interface through .aidl file:

```

// IRemoteService.aidl
package com.example.android;

// Declare any non-default types here with import statements

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();
}

```

Now after build application, sdk tools will generate appropriate .java file. This file will contain **Stub** class which implements our aidl interface, and which we need to extend:

```

public class RemoteService extends Service {

    private final IRemoteService.Stub binder = new IRemoteService.Stub() {
        @Override
        public int getPid() throws RemoteException {
            return Process.myPid();
        }
    };
}

```

```

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return binder;
}
}

```

然后在活动中：

```

public class MainActivity extends AppCompatActivity {
    private final ServiceConnection connection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            IRemoteService service = IRemoteService.Stub.asInterface(iBinder);
            Toast.makeText(this, "活动进程: " + Process.myPid + ", 服务进程: " +
getRemotePid(service), LENGTH_SHORT).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {}
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, RemoteService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unbindService(connection);
    }

    private int getRemotePid(IRemoteService service) {
        int result = -1;

        try {
result = service.getPid();
        } catch (RemoteException e) {
e.printStackTrace();
        }

        return result;
    }
}

```

```

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return binder;
}
}

```

Then in activity:

```

public class MainActivity extends AppCompatActivity {
    private final ServiceConnection connection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            IRemoteService service = IRemoteService.Stub.asInterface(iBinder);
            Toast.makeText(this, "Activity process: " + Process.myPid + ", Service process: " +
getRemotePid(service), LENGTH_SHORT).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {}
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, RemoteService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unbindService(connection);
    }

    private int getRemotePid(IRemoteService service) {
        int result = -1;

        try {
result = service.getPid();
        } catch (RemoteException e) {
e.printStackTrace();
        }

        return result;
    }
}

```

第46章：清单文件

清单文件是一个必须的文件，名称必须为“AndroidManifest.xml”，位于应用程序的根目录。它指定应用名称、图标、Java包名、版本、活动（Activity）、服务（Service）、应用权限及其他信息。

第46.1节：声明组件

清单文件的主要任务是告知系统应用的组件。例如，清单文件可以如下声明一个活动：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
            ...
        </activity>
    ...
    </application>
</manifest>
```

在<application>元素中，android:icon属性指向用于标识应用程序的图标资源。

在该元素中，android:name 属性指定了 Activity 子类的完全限定类名，
android:label 属性指定用于活动的用户可见标签字符串。

您必须以这种方式声明所有应用组件：

- <activity> 元素用于活动
- <service> 元素用于服务
- <receiver> 元素用于广播接收器
- <provider> 元素用于内容提供者

您在源代码中包含但未在清单中声明的活动、服务和内容提供者对系统不可见，因此永远无法运行。然而，广播接收器可以在清单中声明，也可以在代码中动态创建（作为 BroadcastReceiver 对象）并通过调用 registerReceiver()。

有关如何为您的应用构建清单文件的更多信息，请参阅 AndroidManifest.xml 文件文档。

第46.2节：在清单文件中声明权限

您的应用程序若需访问受保护的API部分或与其他应用程序交互，必须在您的AndroidManifest.xml文件中声明所需权限。此操作通过<uses-permission />标签完成。

语法

```
<uses-permission android:name="string"
                 android:maxSdkVersion="integer"/>
```

android:name: 这是所需权限的名称

Chapter 46: The Manifest File

The Manifest is an obligatory file named exactly "AndroidManifest.xml" and located in the app's root directory. It specifies the app name, icon, Java package name, version, declaration of Activities, Services, app permissions and other information.

Section 46.1: Declaring Components

The primary task of the manifest is to inform the system about the app's components. For example, a manifest file can declare an activity as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
            ...
        </activity>
    ...
    </application>
</manifest>
```

In the <application> element, the android:icon attribute points to resources for an icon that identifies the app.

In the element, the android:name attribute specifies the fully qualified class name of the Activity subclass and the android:label attribute specifies a string to use as the user-visible label for the activity.

You must declare all app components this way:

- <activity> elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

Activities, services, and content providers that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as BroadcastReceiver objects) and registered with the system by calling registerReceiver().

For more about how to structure the manifest file for your app, see The AndroidManifest.xml File documentation.

Section 46.2: Declaring permissions in your manifest file

Any permission required by your application to access a protected part of the API or to interact with other applications must be declared in your AndroidManifest.xml file. This is done using the <uses-permission /> tag.

Syntax

```
<uses-permission android:name="string"
                 android:maxSdkVersion="integer"/>
```

android:name: This is the name of the required permission

android:maxSdkVersion: 该权限应授予您的应用的最高API级别。设置此权限是可选的，仅当您的应用所需权限在某个API级别后不再需要时才应设置。

示例 AndroidManifest.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.samplepackage">

    <!-- 请求互联网权限 -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- 请求相机权限 -->
    <uses-permission android:name="android.permission.CAMERA"/>

    <!-- 请求写入外部存储的权限 -->
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="18" />

    <application>....</application>
</manifest>
```

* 另请参见权限主题。

android:maxSdkVersion: The highest API level at which this permission should be granted to your app. Setting this permission is optional and should only be set if the permission your app requires is no longer needed at a certain API level.

Sample AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.samplepackage">

    <!-- request internet permission -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- request camera permission -->
    <uses-permission android:name="android.permission.CAMERA"/>

    <!-- request permission to write to external storage -->
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="18" />

    <application>....</application>
</manifest>
```

* Also see the Permissions topic.

第47章：Android的Gradle

Gradle是一个基于JVM的构建系统，允许开发者编写高级脚本来自动化编译和应用程序生成的过程。它是一个灵活的基于插件的系统，允许你自动化构建过程的各个方面；包括编译和签名一个.jar，下载和管理外部依赖，向AndroidManifest注入字段或使用特定的SDK版本。

第47.1节：一个基本的build.gradle文件

这是一个模块中默认的build.gradle文件示例。

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.3'

    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'keystorePassword'
        }
    }
    defaultConfig {
        applicationId 'com.company.applicationName'
        minSdkVersion 14
        targetSdkVersion 25
        versionCode 1
        versionName '1.0'
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'

    testCompile 'junit:junit:4.12'
}
```

DSL（领域特定语言）

上面文件中的每个块称为DSL（领域专用语言）。

插件

第一行，应用插件：'com.android.application'，应用了Gradle的Android插件到构建中，

Chapter 47: Gradle for Android

Gradle is a JVM-based build system that enables developers to write high-level scripts that can be used to automate the process of compilation and application production. It is a flexible plugin-based system, which allows you to automate various aspects of the build process; including compiling and signing a .jar, downloading and managing external dependencies, injecting fields into the AndroidManifest or utilising specific SDK versions.

Section 47.1: A basic build.gradle file

This is an example of a default build.gradle file in a module.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.3'

    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'keystorePassword'
        }
    }
    defaultConfig {
        applicationId 'com.company.applicationName'
        minSdkVersion 14
        targetSdkVersion 25
        versionCode 1
        versionName '1.0'
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'

    testCompile 'junit:junit:4.12'
}
```

DSL (domain-specific language)

Each block in the file above is called a DSL (domain-specific language).

Plugins

The first line, apply plugin: 'com.android.application', applies the Android plugin for Gradle to the build and

使得`android {}`代码块可用，以声明Android特定的构建选项。

对于一个Android应用程序：

应用插件：`'com.android.application'`

对于一个Android库：

应用插件：`'com.android.library'`

理解上面示例中的DSL

第二部分，`android {...}`代码块，是Android的DSL，包含了关于你的项目的信息。

例如，您可以设置`compileSdkVersion`来指定Android API级别，Gradle应使用该级别来编译您的应用。

子块`defaultConfig`包含您的清单文件的默认设置。您可以通过产品风味（Product Flavors）来覆盖这些设置。

您可以在以下示例中找到更多信息：

- 应用模块的DSL
- 构建类型
- 产品风味
- 签名设置

依赖项

`dependencies`块定义在`android`块之外：这意味着它不是由Android插件定义的，而是标准的Gradle配置。

`dependencies`块指定您希望包含在应用中的外部库（通常是Android库，但Java库也有效）。Gradle会自动为您下载这些依赖项（如果本地没有可用副本），您只需在想添加另一个库时添加类似的`compile`行即可。

让我们来看这里的其中一行：

`compile 'com.android.support:design:25.3.1'`

这行基本上表示

向我的项目添加对 Android 支持设计库的依赖。

Gradle 将确保库被下载并存在，以便您可以在应用中使用它，其代码也将包含在您的应用中。

如果您熟悉 Maven，这种语法是`GroupId`、冒号、`ArtifactId`、另一个冒号，然后是您希望包含的依赖版本，这样您可以完全控制版本管理。

虽然可以使用加号（+）指定工件版本，但最佳做法是避免这样做；如果库在您不知情的情况下更新了破坏性更改，可能会导致您的应用崩溃。

makes the `android {}` block available to declare Android-specific build options.

For an **Android Application**:

`apply plugin: 'com.android.application'`

For an **Android Library**:

`apply plugin: 'com.android.library'`

Understanding the DSLs in the sample above

The second part, The `android {...}` block, is the Android DSL which contains information about your project.

For example, you can set the `compileSdkVersion` which specifies the Android API level , Which should be used by Gradle to compile your app.

The sub-block `defaultConfig` holds the defaults for your manifest. You can override them with Product Flavors.

You can find more info in these examples:

- DSL for the app module
- Build Types
- Product Flavors
- Signing settings

Dependencies

The `dependencies` block is defined outside the `android` block `{...}` : This means it's not defined by the Android plugin but it's standard Gradle.

The `dependencies` block specifies what external libraries (typically Android libraries, but Java libraries are also valid) you wish to include in your app. Gradle will automatically download these dependencies for you (if there is no local copy available), you just need to add similar `compile` lines when you wish to add another library.

Let's look at one of the lines present here:

`compile 'com.android.support:design:25.3.1'`

This line basically says

add a dependency on the Android support design library to my project.

Gradle will ensure that the library is downloaded and present so that you can use it in your app, and its code will also be included in your app.

If you're familiar with Maven, this syntax is the `GroupId`, a colon, `ArtifactId`, another colon, then the version of the dependency you wish to include, giving you full control over versioning.

While it is possible to specify artifact versions using the plus (+) sign, best practice is to avoid doing so; it can lead to issues if the library gets updated with breaking changes without your knowledge, which would likely lead to crashes in your app.

您可以添加不同类型的依赖：

- 本地二进制依赖
- 模块依赖
- 远程依赖

应特别注意 aar 扁平依赖。

您可以在本主题中找到更多详细信息。

关于appcompat-v7中的-v7的说明

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

这仅表示该库（appcompat）兼容 Android API 级别 7 及更高版本。

关于junit:junit:4.12

这是用于单元测试的测试依赖。

指定针对不同构建配置的依赖项

您可以指定某个依赖项仅用于特定的构建配置，或者通过使用debugCompile、testCompile或releaseCompile来为构建类型或产品风味（例如调试、测试或发布）定义不同的依赖项，而不是通常的compile。

这有助于将测试和调试相关的依赖项排除在发布构建之外，从而使您的发布APK尽可能精简，并有助于确保任何调试信息都无法被用来获取有关您的应用的内部信息。

signingConfig

signingConfig允许您配置Gradle以包含keystore信息，并确保使用这些配置构建的APK已签名且准备好发布到Play商店。

您可以在这里找到专门的主题。

注意：不过不建议将签名凭据保存在Gradle文件中。要移除签名配置，只需省略signingConfigs部分即可。

您可以通过不同方式指定它们：

- 存储在外部文件中
- 存储在设置的环境变量中。

有关更多详细信息，请参见此主题：在不暴露keystore密码的情况下签名APK。

您可以在专门的Gradle主题中找到有关Android的更多信息。

第47.2节：定义和使用构建配置字段

BuildConfigField

You can add different kind of dependencies:

- local binary dependencies
- module dependencies
- remote dependencies

A particular attention should be dedicated to the aar flat dependencies.

You can find more details in this topic.

Note about the **-v7 in appcompat-v7**

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

This simply means that this **library** (appcompat) is compatible with the Android API level 7 and forward.

Note about the **junit:junit:4.12**

This is Testing dependency for Unit testing.

Specifying dependencies specific to different build configurations

You can specify that a dependency should only be used for a certain build configuration or you can define different dependencies for the build types or the product flavors (e.g., debug, test or release) by using debugCompile, testCompile or releaseCompile instead of the usual compile.

This is helpful for keeping test- and debug- related dependencies out of your release build, which will keep your release APK as slim as possible and help to ensure that any debug information cannot be used to obtain internal information about your app.

signingConfig

The signingConfig allows you to configure your Gradle to include keystore information and ensure that the APK built using these configurations are signed and ready for Play Store release.

Here you can find a dedicated topic.

Note: It's not recommended though to keep the signing credentials inside your Gradle file. To remove the signing configurations, just omit the signingConfigs portion.

You can specify them in different ways:

- storing in an external file
- storing them in setting environment variables.

See this topic for more details : Sign APK without exposing keystore password.

You can find further information about Gradle for Android in the dedicated Gradle topic.

Section 47.2: Define and use Build Configuration Fields

BuildConfigField

Gradle允许使用buildConfigField行来定义常量。这些常量将在运行时作为BuildConfig类的静态字段访问。通过在defaultConfig块中定义所有字段，然后根据需要为各个构建风味覆盖它们，可以用来创建不同的风味。

此示例定义了构建日期，并将构建标记为生产而非测试：

```
android {  
    ...  
    defaultConfig {  
        ...  
        // 定义构建日期  
        buildConfigField "long", "BUILD_DATE", System.currentTimeMillis() + "L"  
        // 定义此构建是否为生产构建  
        buildConfigField "boolean", "IS_PRODUCTION", "false"  
        // 注意定义字符串时需要转义  
        buildConfigField "String", "API_KEY", "\"my_api_key\""  
    }  
  
    productFlavors {  
        prod {  
            // 覆盖 flavor "prod" 的生产标志  
            buildConfigField "boolean", "IS_PRODUCTION", "true"  
            resValue 'string', 'app_name', 'My App Name'  
        }  
        dev {  
            // 继承默认字段  
            resValue 'string', 'app_name', 'My App Name - Dev'  
        }  
    }  
}
```

自动生成的 <package_name>.BuildConfig.java 文件位于 gen 文件夹中，包含基于上述指令的以下字段：

```
public class BuildConfig {  
    // ... 其他生成的字段 ...  
    public static final long BUILD_DATE = 1469504547000L;  
    public static final boolean IS_PRODUCTION = false;  
    public static final String API_KEY = "my_api_key";  
}
```

现在可以通过访问生成的BuildConfig类，在应用运行时使用定义的字段：

```
public void example() {  
    // 格式化构建日期  
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");  
    String buildDate = dateFormat.format(new Date(BuildConfig.BUILD_DATE));  
    Log.d("build date", buildDate);  
  
    // 根据是否为生产版本执行操作  
    if (BuildConfig.IS_PRODUCTION) {  
        connectToProductionApiEndpoint();  
    } else {  
        connectToStagingApiEndpoint();  
    }  
}
```

ResValue

Gradle allows buildConfigField lines to define constants. These constants will be accessible at runtime as static fields of the BuildConfig class. This can be used to create flavors by defining all fields within the defaultConfig block, then overriding them for individual build flavors as needed.

This example defines the build date and flags the build for production rather than test:

```
android {  
    ...  
    defaultConfig {  
        ...  
        // defining the build date  
        buildConfigField "long", "BUILD_DATE", System.currentTimeMillis() + "L"  
        // define whether this build is a production build  
        buildConfigField "boolean", "IS_PRODUCTION", "false"  
        // note that to define a string you need to escape it  
        buildConfigField "String", "API_KEY", "\"my_api_key\""  
    }  
  
    productFlavors {  
        prod {  
            // override the productive flag for the flavor "prod"  
            buildConfigField "boolean", "IS_PRODUCTION", "true"  
            resValue 'string', 'app_name', 'My App Name'  
        }  
        dev {  
            // inherit default fields  
            resValue 'string', 'app_name', 'My App Name - Dev'  
        }  
    }  
}
```

The automatically-generated <package_name>.BuildConfig.java in the gen folder contains the following fields based on the directive above:

```
public class BuildConfig {  
    // ... other generated fields ...  
    public static final long BUILD_DATE = 1469504547000L;  
    public static final boolean IS_PRODUCTION = false;  
    public static final String API_KEY = "my_api_key";  
}
```

The defined fields can now be used within the app at runtime by accessing the generated BuildConfig class:

```
public void example() {  
    // format the build date  
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");  
    String buildDate = dateFormat.format(new Date(BuildConfig.BUILD_DATE));  
    Log.d("build date", buildDate);  
  
    // do something depending whether this is a productive build  
    if (BuildConfig.IS_PRODUCTION) {  
        connectToProductionApiEndpoint();  
    } else {  
        connectToStagingApiEndpoint();  
    }  
}
```

ResValue

productFlavors 中的 resValue 创建了一个资源值。它可以是任何类型的资源 (string、dimen、color 等)。这类似于在相应的文件中定义资源：例如，在 strings.xml 文件中定义字符串。其优点是 gradle 中定义的资源可以根据你的 productFlavor/buildVariant 进行修改。要访问该值，写的代码与访问资源文件中的资源相同：

```
getResources().getString(R.string.app_name)
```

重要的是，以这种方式定义的资源不能修改文件中已定义的现有资源。它们只能创建新的资源值。

一些库（例如 Google Maps Android API）需要在清单文件中以 meta-data 标签提供 API 密钥。如果调试和生产构建需要不同的密钥，则指定由 Gradle 填充的清单占位符。

在你的 AndroidManifest.xml 文件中：

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="${MAPS_API_KEY}" />
```

然后在你的 build.gradle 文件中相应地设置该字段：

```
android {  
    defaultConfig {  
        ...  
        // Your development key  
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]  
    }  
  
    productFlavors {  
        prod {  
            // Your production key  
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]  
        }  
    }  
}
```

Android 构建系统会自动生成多个字段，并将它们放置在 BuildConfig.java 中。这些字段包括：

字段	描述
DEBUG	一个 Boolean，表示应用程序是处于调试模式还是发布模式
APPLICATION_ID	一个 String，包含应用程序的 ID（例如 com.example.app）
BUILD_TYPE	一个 String，包含应用程序的构建类型（通常是 debug 或 release）
FLAVOR	一个 String，包含构建的特定风味
版本代码	一个包含版本（构建）号的 int 类型。 这与 build.gradle 中的 versionCode 或 AndroidManifest.xml 中的 versionCode 相同。
版本名称	一个包含版本（构建）名称的 String 类型。 这与 build.gradle 中的 versionName 或 AndroidManifest.xml 中的 versionName 相同。

除了上述内容，如果你定义了多个维度的风味（flavor），那么每个维度都会有自己的值。例如，如果你有两个风味维度，分别是 color 和 size，那么你还会以下变量：

字段	描述
----	----

The resValue in the productFlavors creates a resource value. It can be any type of resource (string, dimen, color, etc.). This is similar to defining a resource in the appropriate file: e.g. defining string in a strings.xml file. The advantage being that the one defined in gradle can be modified based on your productFlavor/buildVariant. To access the value, write the same code as if you were accessing a res from the resources file:

```
getResources().getString(R.string.app_name)
```

The important thing is that resources defined this way cannot modify existing resources defined in files. They can only create new resource values.

Some libraries (such as the Google Maps Android API) require an API key provided in the Manifest as a meta-data tag. If different keys are needed for debugging and production builds, specify a manifest placeholder filled in by Gradle.

In your AndroidManifest.xml file:

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="${MAPS_API_KEY}" />
```

And then set the field accordingly in your build.gradle file:

```
android {  
    defaultConfig {  
        ...  
        // Your development key  
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]  
    }  
  
    productFlavors {  
        prod {  
            // Your production key  
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]  
        }  
    }  
}
```

The Android build system generates a number of fields automatically and places them in BuildConfig.java. These fields are:

Field	Description
DEBUG	a Boolean stating if the app is in debug or release mode
APPLICATION_ID	a String containing the ID of the application (e.g. com.example.app)
BUILD_TYPE	a String containing the build type of the application (usually either debug or release)
FLAVOR	a String containing the particular flavor of the build
VERSION_CODE	an int containing the version (build) number. This is the same as versionCode in build.gradle or versionCode in AndroidManifest.xml
VERSION_NAME	a String containing the version (build) name. This is the same as versionName in build.gradle or versionName in AndroidManifest.xml

In addition to the above, if you have defined multiple dimensions of flavor then each dimension will have its own value. For example, if you had two dimensions of flavor for color and size you will also have the following variables:

Field	Description
-------	-------------

FLAVOR_color 一个包含“color”风味值的String类型。

FLAVOR_size一个包含“size”风味值的String类型。

第47.3节：通过“dependencies.gradle”文件集中管理依赖

在处理多模块项目时，将依赖集中在一个位置比分散在多个构建文件中更为方便，尤其是对于常用库，如Android支持库和Firebase库。

一种推荐的方法是分离Gradle构建文件，每个模块一个build.gradle文件，项目根目录一个，以及一个用于依赖管理的文件，例如：

根目录

```
+-- gradleScript/  
|   dependencies.gradle  
+-- module1/  
|   build.gradle  
+-- module2/  
|   build.gradle  
+- build.gradle
```

然后，所有的依赖项都可以在 gradleScript/dependencies.gradle 中找到：

```
ext {  
    // 版本  
    supportVersion = '24.1.0'  
  
    // 支持库依赖  
    supportDependencies = [  
        design: "com.android.support:design:${supportVersion}",  
        recyclerView: "com.android.support:recyclerview-v7:${supportVersion}",  
        cardView: "com.android.support:cardview-v7:${supportVersion}",  
        appCompat: "com.android.support:appcompat-v7:${supportVersion}",  
        supportAnnotation: "com.android.support:support-annotations:${supportVersion}"  
    ]  
  
    firebaseVersion = '9.2.0';  
  
    firebaseDependencies = [  
        core: "com.google.firebaseio:firebase-core:${firebaseVersion}",  
        database: "com.google.firebaseio:firebase-database:${firebaseVersion}",  
        storage: "com.google.firebaseio:firebase-storage:${firebaseVersion}",  
        crash: "com.google.firebaseio:firebase-crash:${firebaseVersion}",  
        auth: "com.google.firebaseio:firebase-auth:${firebaseVersion}",  
        messaging: "com.google.firebaseio:firebase-messaging:${firebaseVersion}",  
        remoteConfig: "com.google.firebaseio:firebase-config:${firebaseVersion}",  
        invites: "com.google.firebaseio:firebase-invites:${firebaseVersion}",  
        adMod: "com.google.firebaseio:firebase-ads:${firebaseVersion}",  
        appIndexing: "com.google.android.gms:play-services-appindexing:${firebaseVersion}"  
    ];  
}
```

然后可以在顶层文件build.gradle中通过该文件应用，如下所示：

```
// 加载依赖  
apply from: 'gradleScript/dependencies.gradle'
```

FLAVOR_color a [String](#) containing the value for the 'color' flavor.

FLAVOR_size a [String](#) containing the value for the 'size' flavor.

Section 47.3: Centralizing dependencies via “dependencies.gradle” file

When working with multi-module projects, it is helpful to centralize dependencies in a single location rather than having them spread across many build files, especially for common libraries such as the Android support libraries and the Firebase libraries.

One recommended way is to separate the Gradle build files, with one `build.gradle` per module, as well as one in the project root and another one for the dependencies, for example:

```
root  
+-- gradleScript/  
|   dependencies.gradle  
+-- module1/  
|   build.gradle  
+-- module2/  
|   build.gradle  
+- build.gradle
```

Then, all of your dependencies can be located in `gradleScript/dependencies.gradle`:

```
ext {  
    // Version  
    supportVersion = '24.1.0'  
  
    // Support Libraries dependencies  
    supportDependencies = [  
        design: "com.android.support:design:${supportVersion}",  
        recyclerView: "com.android.support:recyclerview-v7:${supportVersion}",  
        cardView: "com.android.support:cardview-v7:${supportVersion}",  
        appCompat: "com.android.support:appcompat-v7:${supportVersion}",  
        supportAnnotation: "com.android.support:support-annotations:${supportVersion}"  
    ]  
  
    firebaseVersion = '9.2.0';  
  
    firebaseDependencies = [  
        core: "com.google.firebaseio:firebase-core:${firebaseVersion}",  
        database: "com.google.firebaseio:firebase-database:${firebaseVersion}",  
        storage: "com.google.firebaseio:firebase-storage:${firebaseVersion}",  
        crash: "com.google.firebaseio:firebase-crash:${firebaseVersion}",  
        auth: "com.google.firebaseio:firebase-auth:${firebaseVersion}",  
        messaging: "com.google.firebaseio:firebase-messaging:${firebaseVersion}",  
        remoteConfig: "com.google.firebaseio:firebase-config:${firebaseVersion}",  
        invites: "com.google.firebaseio:firebase-invites:${firebaseVersion}",  
        adMod: "com.google.firebaseio:firebase-ads:${firebaseVersion}",  
        appIndexing: "com.google.android.gms:play-services-appindexing:${firebaseVersion}"  
    ];  
}
```

Which can then be applied from that file in the top level file `build.gradle` like so:

```
// Load dependencies  
apply from: 'gradleScript/dependencies.gradle'
```

在module1/build.gradle中如下使用：

```
// 模块构建文件  
dependencies {  
    // ...  
    compile supportDependencies.appcompat  
    compile supportDependencies.design  
    compile firebaseDependencies.crash  
}
```

另一种方法

通过将版本号声明为变量一次，并在各处使用，可以实现一种更简洁的集中管理库依赖版本的方法。

在工作区根目录的build.gradle文件中添加以下内容：

```
ext.v = [  
    supportVersion: '24.1.1',  
]
```

并且在每个使用相同库的模块中添加所需的库

```
compile "com.android.support:support-v4:${v.supportVersion}"  
compile "com.android.support:recyclerview-v7:${v.supportVersion}"  
compile "com.android.support:design:${v.supportVersion}"  
compile "com.android.support:support-annotations:${v.supportVersion}"
```

第47.4节：签署APK而不暴露密钥库密码

你可以在build.gradle文件中使用以下属性定义签名配置来签署apk：

- storeFile : 密钥库文件
- storePassword: 密钥库密码
- keyAlias: 密钥别名
- keyPassword: 密钥别名密码

在许多情况下，你可能需要避免在build.gradle文件中包含此类信息。

方法A：使用keystore.properties文件配置发布签名

可以配置你的应用的build.gradle，使其从类似keystore.properties的属性文件中读取签名配置信息。

这样设置签名的好处是：

- 你的签名配置信息与build.gradle文件分开
- 你无需在签名过程中手动输入密钥库文件的密码
- 你可以轻松地将keystore.properties文件排除在版本控制之外

首先，在项目根目录下创建一个名为keystore.properties的文件，内容如下（将值替换为自己的）：

```
storeFile=keystore.jks  
storePassword=storePassword  
keyAlias=keyAlias
```

and in the module1/build.gradle like so:

```
// Module build file  
dependencies {  
    // ...  
    compile supportDependencies.appcompat  
    compile supportDependencies.design  
    compile firebaseDependencies.crash  
}
```

Another approach

A less verbose approach for centralizing library dependencies versions can be achieved by declaring the version number as a variable once, and using it everywhere.

In the workspace root build.gradle add this:

```
ext.v = [  
    supportVersion: '24.1.1',  
]
```

And in every module that uses the same library add the needed libraries

```
compile "com.android.support:support-v4:${v.supportVersion}"  
compile "com.android.support:recyclerview-v7:${v.supportVersion}"  
compile "com.android.support:design:${v.supportVersion}"  
compile "com.android.support:support-annotations:${v.supportVersion}"
```

Section 47.4: Sign APK without exposing keystore password

You can define the signing configuration to sign the apk in the build.gradle file using these properties:

- storeFile : the keystore file
- storePassword: the keystore password
- keyAlias: a key alias name
- keyPassword: A key alias password

In many case you may need to avoid this kind of info in the build.gradle file.

Method A: Configure release signing using a keystore.properties file

It's possible to configure your app's build.gradle so that it will read your signing configuration information from a properties file like keystore.properties.

Setting up signing like this is beneficial because:

- Your signing configuration information is separate from your build.gradle file
- You do not have to intervene during the signing process in order to provide passwords for your keystore file
- You can easily exclude the keystore.properties file from version control

First, create a file called keystore.properties in the root of your project with content like this (replacing the values with your own):

```
storeFile=keystore.jks  
storePassword=storePassword  
keyAlias=keyAlias
```

```
keyPassword=keyPassword
```

现在，在你的应用的build.gradle文件中，按如下方式设置signingConfigs块：

```
signingConfigs {
release {
def propsFile = rootProject.file('keystore.properties')
if (propsFile.exists()) {
def props = new Properties()
props.load(new FileInputStream(propsFile))
storeFile = file(props['storeFile'])
storePassword = props['storePassword']
keyAlias = props['keyAlias']
keyPassword = props['keyPassword']
}
}
}
```

这就是全部内容，但别忘了将你的密钥库文件和 keystore.properties 文件从版本控制中排除。

需要注意的几点：

- keystore.properties 文件中指定的 storeFile 路径应相对于你的应用的 build.gradle 文件。本示例假设密钥库文件与应用的 build.gradle 文件在同一目录下。
- 本示例将 keystore.properties 文件放在项目根目录。如果你放在其他位置，请确保将 rootProject.file('keystore.properties') 中的值更改为相对于项目根目录的你的文件位置。

方法二：通过使用环境变量

同样的效果也可以在没有属性文件的情况下实现，使密码更难被发现：

```
android {
signingConfigs {
release {
storeFile file('/your/keystore/location/key')
keyAlias 'your_alias'
String ps = System.getenv("ps")
if (ps == null) {
throw new GradleException('缺少 ps 环境变量')
}
keyPassword ps
storePassword ps
}
}
}
```

"ps"环境变量可以是全局的，但更安全的方法是仅将其添加到 Android Studio 的 shell 中。

在 Linux 中，可以通过编辑 Android Studio 的Desktop Entry 来实现

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

```
keyPassword=keyPassword
```

Now, in your app's build.gradle file, set up the signingConfigs block as follows:

```
android {
...
signingConfigs {
release {
def propsFile = rootProject.file('keystore.properties')
if (propsFile.exists()) {
def props = new Properties()
props.load(new FileInputStream(propsFile))
storeFile = file(props['storeFile'])
storePassword = props['storePassword']
keyAlias = props['keyAlias']
keyPassword = props['keyPassword']
}
}
}
}
```

That's really all there is to it, **but don't forget to exclude both your keystore file and your keystore.properties file from version control.**

A couple of things to note:

- The storeFile path specified in the keystore.properties file should be relative to your app's build.gradle file. This example assumes that the keystore file is in the same directory as the app's build.gradle file.
- This example has the keystore.properties file in the root of the project. If you put it somewhere else, be sure to change the value in rootProject.file('keystore.properties') to the location of yours, relative to the root of your project.

Method B: By using an environment variable

The same can be achieved also without a properties file, making the password harder to find:

```
android {
signingConfigs {
release {
storeFile file('/your/keystore/location/key')
keyAlias 'your_alias'
String ps = System.getenv("ps")
if (ps == null) {
throw new GradleException('missing ps env variable')
}
keyPassword ps
storePassword ps
}
}
}
```

The "ps" environment variable can be global, but a safer approach can be by adding it to the shell of Android Studio only.

In linux this can be done by editing Android Studio's Desktop Entry

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

您可以在本主题中找到更多详细信息。

第47.5节：添加产品口味特定的依赖项

依赖项可以针对特定的产品风味添加，类似于针对特定的构建配置添加依赖项。

在此示例中，假设我们已经定义了两个产品风味，分别称为free和paid（关于定义风味的更多内容见[此处](#)）。

然后我们可以为free风味添加AdMob依赖，为paid风味添加Picasso库，示例如下：

```
android {  
    ...  
  
    productFlavors {  
        free {  
            applicationId "com.example.app.free"  
            versionName "1.0-free"  
        }  
        paid {  
            applicationId "com.example.app.paid"  
            versionName "1.0-paid"  
        }  
    }  
  
    ...  
    dependencies {  
        ...  
        // 仅为free风味添加AdMob  
        freeCompile 'com.android.support:appcompat-v7:23.1.1'  
        freeCompile 'com.google.android.gms:play-services-ads:8.4.0'  
        freeCompile 'com.android.support:support-v4:23.1.1'  
  
        // 仅为付费版本添加 picasso  
        paidCompile 'com.squareup.picasso:picasso:2.5.2'  
    }  
}
```

You can find more details in this topic.

Section 47.5: Adding product flavor-specific dependencies

Dependencies can be added for a specific product flavor, similar to how they can be added for specific build configurations.

For this example, assume that we have already defined two product flavors called free and paid (more on defining flavors [here](#)).

We can then add the AdMob dependency for the free flavor, and the Picasso library for the paid one like so:

```
android {  
    ...  
  
    productFlavors {  
        free {  
            applicationId "com.example.app.free"  
            versionName "1.0-free"  
        }  
        paid {  
            applicationId "com.example.app.paid"  
            versionName "1.0-paid"  
        }  
    }  
  
    ...  
    dependencies {  
        ...  
        // Add AdMob only for free flavor  
        freeCompile 'com.android.support:appcompat-v7:23.1.1'  
        freeCompile 'com.google.android.gms:play-services-ads:8.4.0'  
        freeCompile 'com.android.support:support-v4:23.1.1'  
  
        // Add picasso only for paid flavor  
        paidCompile 'com.squareup.picasso:picasso:2.5.2'  
    }  
}
```

第47.6节：为构建类型和产品风味指定不同的应用程序ID

您可以使用

applicationIdSuffix配置属性为每个buildType或productFlavor指定不同的应用程序ID或包名：

为每个buildType添加applicationId后缀的示例：

```
defaultConfig {  
    applicationId "com.package.android"  
    minSdkVersion 17  
    targetSdkVersion 23  
    versionCode 1  
    versionName "1.0"  
}  
  
buildTypes {  
    release {  
        debuggable false  
    }
```

Section 47.6: Specifying different application IDs for build types and product flavors

You can specify different application IDs or package names for each buildType or productFlavor using the **applicationIdSuffix** configuration attribute:

Example of suffixing the applicationId for each buildType:

```
defaultConfig {  
    applicationId "com.package.android"  
    minSdkVersion 17  
    targetSdkVersion 23  
    versionCode 1  
    versionName "1.0"  
}  
  
buildTypes {  
    release {  
        debuggable false  
    }
```

```

}
development {
    debuggable true
    applicationIdSuffix ".dev"
}

testing {
    debuggable true
    applicationIdSuffix ".qa"
}

```

我们最终的applicationIds将会是：

- com.package.android 用于release
- com.package.android.**dev** 用于development
- com.package.android.**qa** 用于testing

这也应用于productFlavors：

```

productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}

```

生成的applicationIds将会是：

- com.package.android.**free** 对应free 版本
- com.package.android.**paid** 对应paid 版本

第47.7节：通过 "version.properties"

你可以使用Gradle在每次构建时自动递增包版本号。为此，请创建一个 `version.properties` 文件，放在与你的 `build.gradle` 同一目录下，内容如下：

```

VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1

```

(根据需要更改主版本号和次版本号的值)。然后在你的 `build.gradle` 文件的

`android` 部分添加以下代码：

```

// 从本地文件读取版本信息并适当递增
def versionPropsFile = file('version.properties')
if (versionPropsFile.canRead()) {
    def Properties versionProps = new Properties()

    versionProps.load(new FileInputStream(versionPropsFile))

    def versionMajor = versionProps['VERSION_MAJOR'].toInteger()
    def versionMinor = versionProps['VERSION_MINOR'].toInteger()
}

```

```

}
development {
    debuggable true
    applicationIdSuffix ".dev"
}

testing {
    debuggable true
    applicationIdSuffix ".qa"
}

```

Our resulting applicationIds would now be:

- com.package.android for release
- com.package.android.**dev** for development
- com.package.android.**qa** for testing

This can be done for productFlavors as well:

```

productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}

```

The resulting applicationIds would be:

- com.package.android.**free** for the free flavor
- com.package.android.**paid** for the paid flavor

Section 47.7: Versioning your builds via "version.properties" file

You can use Gradle to auto-increment your package version each time you build it. To do so create a `version.properties` file in the same directory as your `build.gradle` with the following contents:

```

VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1

```

(Changing the values for major and minor as you see fit). Then in your `build.gradle` add the following code to the `android` section:

```

// Read version information from local file and increment as appropriate
def versionPropsFile = file('version.properties')
if (versionPropsFile.canRead()) {
    def Properties versionProps = new Properties()

    versionProps.load(new FileInputStream(versionPropsFile))

    def versionMajor = versionProps['VERSION_MAJOR'].toInteger()
    def versionMinor = versionProps['VERSION_MINOR'].toInteger()
}

```

```

def versionBuild = versionProps['VERSION_BUILD'].toInteger() + 1

// 更新本地文件中的构建号
versionProps['VERSION_BUILD'] = versionBuild.toString()
versionProps.store(versionPropsFile.newWriter(), null)

defaultConfig {
    versionCode versionBuild
versionName "${versionMajor}.${versionMinor}." + String.format("%05d", versionBuild)
}
}

```

该信息可以在Java中通过字符串 `BuildConfig.VERSION_NAME` 访问，获取完整的 `{major}.{minor}.{build}` 版本号，通过整数 `BuildConfig.VERSION_CODE` 访问，仅获取构建号。

第47.8节：定义产品风味

产品风味在 `build.gradle` 文件中的 `android { ... }` 块内定义，如下所示。

```

...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}

```

通过这样做，我们现在有了两种额外的产品版本：免费和付费。每个版本都可以有其特定的配置和属性。例如，我们的两个新版本都有一个独立的 `applicationId` 和 `versionName`，与我们现有的 `main` 版本（默认可用，因此此处未显示）不同。

第47.9节：更改输出apk名称并添加版本名称：

这是更改输出应用程序文件名 (.apk) 的代码。名称可以通过为 `newName` 赋予不同的值来配置

```

android {

    applicationVariants.all { variant ->
        def newName = "ApkName";
        variant.outputs.each { output ->
            def apk = output.outputFile;

            newName += "-v" + defaultConfig.versionName;
            if (variant.buildType.name == "release") {
                newName += "-release.apk";
            } else {
                newName += ".apk";
            }
            if (!output.zipAlign) {

```

```

def versionBuild = versionProps['VERSION_BUILD'].toInteger() + 1

// Update the build number in the local file
versionProps['VERSION_BUILD'] = versionBuild.toString()
versionProps.store(versionPropsFile.newWriter(), null)

defaultConfig {
    versionCode versionBuild
    versionName "${versionMajor}.${versionMinor}." + String.format("%05d", versionBuild)
}
}

```

The information can be accessed in Java as a string `BuildConfig.VERSION_NAME` for the complete `{major}.{minor}.{build}` number and as an integer `BuildConfig.VERSION_CODE` for just the build number.

Section 47.8: Defining product flavors

Product flavors are defined in the `build.gradle` file inside the `android { ... }` block as seen below.

```

...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}

```

By doing this, we now have two additional product flavors: `free` and `paid`. Each can have its own specific configuration and attributes. For example, both of our new flavors has a separate `applicationId` and `versionName` than our existing `main` flavor (available by default, so not shown here).

Section 47.9: Changing output apk name and add version name:

This is the code for changing output application file name (.apk). The name can be configured by assigning a different value to `newName`

```

android {

    applicationVariants.all { variant ->
        def newName = "ApkName";
        variant.outputs.each { output ->
            def apk = output.outputFile;

            newName += "-v" + defaultConfig.versionName;
            if (variant.buildType.name == "release") {
                newName += "-release.apk";
            } else {
                newName += ".apk";
            }
            if (!output.zipAlign) {

```

```

newName = newName.replace(".apk", "-unaligned.apk");
}

output.outputFile = new File(apk.parentFile, newName);
logger.info("INFO: Set outputFile to "
+ outputFile
+ " 用于 [" + output.name + "]");
}
}

```

```

newName = newName.replace(".apk", "-unaligned.apk");
}

output.outputFile = new File(apk.parentFile, newName);
logger.info("INFO: Set outputFile to "
+ outputFile
+ " 用于 [" + output.name + "]");
}
}

```

第47.10节：添加产品风味特定资源

可以为特定的产品风味添加资源。

在此示例中，假设我们已经定义了两个产品风味，分别称为free和paid。为了添加产品风味特定资源，我们在main/res文件夹旁边创建额外的资源文件夹，然后像平常一样向其中添加资源。此示例中，我们将为每个产品风味定义一个字符串status：

/src/main/res/values/strings.xml

```

<resources>
    <string name="status">默认</string>
</resources>

```

/src/free/res/values/strings.xml

```

<resources>
    <string name="status">免费</string>
</resources>

```

/src/paid/res/values/strings.xml

```

<resources>
    <string name="status">付费</string>
</resources>

```

产品特定口味的status字符串将覆盖main口味中status的值。

第47.11节：为什么Android Studio项目中有两个build.gradle文件？

<PROJECT_ROOT>\app\build.gradle是针对app模块的。

<PROJECT_ROOT>\build.gradle是一个“顶层构建文件”，你可以在这里添加所有子项目/模块共有的配置选项。

如果你在项目中使用了另一个模块，作为本地库，你会有另一个build.gradle文件：

<PROJECT_ROOT>\module\build.gradle

在顶层文件中，你可以指定公共属性，比如buildscript块或一些公共属性。

```

buildscript {
    repositories {
        mavenCentral()
    }
}

```

Section 47.10: Adding product flavor-specific resources

Resources can be added for a specific product flavor.

For this example, assume that we have already defined two product flavors called free and paid. In order to add product flavor-specific resources, we create additional resource folders alongside the main/res folder, which we can then add resources to like usual. For this example, we'll define a string, status, for each product flavor:

/src/main/res/values/strings.xml

```

<resources>
    <string name="status">Default</string>
</resources>

```

/src/free/res/values/strings.xml

```

<resources>
    <string name="status">Free</string>
</resources>

```

/src/paid/res/values/strings.xml

```

<resources>
    <string name="status">Paid</string>
</resources>

```

The product flavor-specific status strings will override the value for status in the main flavor.

Section 47.11: Why are there two build.gradle files in an Android Studio project?

<PROJECT_ROOT>\app\build.gradle is specific for **app module**.

<PROJECT_ROOT>\build.gradle is a "**Top-level build file**" where you can add configuration options common to all sub-projects/modules.

If you use another module in your project, as a local library you would have another build.gradle file:
<PROJECT_ROOT>\module\build.gradle

In the top level file you can specify common properties as the buildscript block or some common properties.

```

buildscript {
    repositories {
        mavenCentral()
    }
}

```

```

dependencies {
    classpath 'com.android.tools.build:gradle:2.2.0'
        classpath 'com.google.gms:google-services:3.0.0'
    }
}

ext {
    compileSdkVersion = 23
    buildToolsVersion = "23.0.1"
}

```

在app\build.gradle中，您只需为模块定义属性：

应用插件：'com.android.application'

```

android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //....
}

```

第47.12节：针对特定风味资源的目录结构

应用构建的不同风味可以包含不同的资源。要创建特定风味的资源，请在 src目录下创建一个以风味名称小写命名的目录，并以通常的方式添加资源。

例如，如果您有一个名为Development的风味，并且想为其提供一个独特的启动器图标，您可以创建目录 src/development/res/drawable-mdpi，并在该目录中创建一个带有开发专用图标的ic_launcher.png文件。

目录结构如下所示：

```

src/
main/
res/
drawable-mdpi/
ic_launcher.png <- 默认启动器图标
    开发/
    res/
    drawable-mdpi/
    ic_launcher.png <- 当产品风味为'开发'时使用的启动器图标

```

(当然，在这种情况下，你还需要为 drawable-hdpi、drawable-xhdpi 等创建图标)。

第47.13节：使用gradle启用Proguard

要为你的应用启用Proguard配置，需要在模块级gradle文件中启用它。你需要将minifyEnabled的值设置为true。

```

buildTypes {
    release {

```

```

        dependencies {
            classpath 'com.android.tools.build:gradle:2.2.0'
            classpath 'com.google.gms:google-services:3.0.0'
        }
    }

    ext {
        compileSdkVersion = 23
        buildToolsVersion = "23.0.1"
    }

```

In the app\build.gradle you define only the properties for the module:

apply plugin: 'com.android.application'

```

android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //....
}

```

Section 47.12: Directory structure for flavor-specific resources

Different flavors of application builds can contain different resources. To create a flavor-specific resource make a directory with the lower-case name of your flavor in the src directory and add your resources in the same way you would normally.

For example, if you had a flavour Development and wanted to provide a distinct launcher icon for it you would create a directory src/development/res/drawable-mdpi and inside that directory create an ic_launcher.png file with your development-specific icon.

The directory structure will look like this:

```

src/
    main/
        res/
            drawable-mdpi/
                ic_launcher.png <- the default launcher icon
    development/
        res/
            drawable-mdpi/
                ic_launcher.png <- the launcher icon used when the product flavor is 'Development'

```

(Of course, in this case you would also create icons for drawable-hdpi, drawable-xhdpi etc).

Section 47.13: Enable Proguard using gradle

For enabling Proguard configurations for your application you need to enable it in your module-level gradle file. You need to set the value of minifyEnabled to true.

```

buildTypes {
    release {

```

```
minifyEnabled true  
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
}
```

上述代码将应用默认Android SDK中包含的Proguard配置，结合你模块中的"proguard-rules.pro"文件，作用于你的发布apk。

第47.14节：忽略构建变体

出于某些原因，你可能想忽略构建变体。例如：你有一个'mock'产品风味，你仅将其用于调试目的，如单元测试/仪器测试。

让我们忽略项目中的mockRelease变体。打开build.gradle文件并写入：

```
// 删除 mockRelease, 因为不需要它。  
android.variantFilter { variant ->  
    如果 (variant.buildType.name.equals('release') &&  
variant.getFlavors().get(0).name.equals('mock')) {  
        variant.setIgnore(true);  
    }  
}
```

第47.15节：为Gradle和Android Studio启用实验性的NDK插件支持

启用并配置实验性的Gradle插件，以提升Android Studio对NDK的支持。请确认您满足以下要求：

- Gradle 2.10（本示例使用）
- Android NDK r10或更高版本
- Android SDK，构建工具版本19.0.0或更高

配置 MyApp/build.gradle 文件

编辑 build.gradle 中的 dependencies.classpath 行，例如：

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

到

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 是撰写时的最新版本。请自行检查最新版本并相应调整您的代码行)

build.gradle 文件应类似如下：

```
buildscript {  
    repositories {  
        jcenter()  
    }  
dependencies {  
    classpath 'com.android.tools.build:gradle-experimental:0.7.2'  
}
```

```
minifyEnabled true  
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
}
```

The above code will apply your Proguard configurations contained in the default Android SDK combined with the "proguard-rules.pro" file on your module to your released apk.

Section 47.14: Ignoring build variant

For some reasons you may want to ignore your build variants. For example: you have 'mock' product flavour and you use it only for debug purposes, such as unit/instrumentation tests.

Let's ignore **mockRelease** variant from our project. Open **build.gradle** file and write:

```
// Remove mockRelease as it's not needed.  
android.variantFilter { variant ->  
    if (variant.buildType.name.equals('release') &&  
variant.getFlavors().get(0).name.equals('mock')) {  
        variant.setIgnore(true);  
    }  
}
```

Section 47.15: Enable experimental NDK plugin support for Gradle and AndroidStudio

Enable and configure the experimental Gradle plugin to improve AndroidStudio's NDK support. Check that you fulfill the following requirements:

- Gradle 2.10 (for this example)
- Android NDK r10 or later
- Android SDK with build tools v19.0.0 or later

Configure MyApp/build.gradle file

Edit the dependencies.classpath line in build.gradle from e.g.

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

to

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 was the latest version at the time of writing. Check the latest version yourself and adapt your line accordingly)

The build.gradle file should look similar to this:

```
buildscript {  
    repositories {  
        jcenter()  
    }  
dependencies {  
    classpath 'com.android.tools.build:gradle-experimental:0.7.2'  
}
```

```

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}

```

配置 MyApp/app/build.gradle 文件

编辑 build.gradle 文件，使其类似以下示例。您的版本号可能不同。

```

apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion 19
        buildToolsVersion "24.0.1"

        defaultConfig {
            applicationId "com.example.mydomain.myapp"
            minSdkVersion.apiLevel 19
            targetSdkVersion.apiLevel 19
            versionCode 1
            versionName "1.0"
        }
        buildTypes {
            release {
                minifyEnabled false
                proguardFiles.add(file('proguard-android.txt'))
            }
        }
    ndk {
        moduleName "myLib"
    }

    /* 以下几行是一些可选标志的示例,
       您可以设置这些标志来配置您的构建环境
    */
    cppFlags.add("-I${file("path/to/my/includes/dir")}.toString())
    cppFlags.add("-std=c++11")
    ldLibs.addAll(['log', 'm'])
    stl = "c++_static"
    abiFilters.add("armeabi-v7a")
}
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}

```

同步并检查Gradle文件中是否有错误，然后再继续。

测试插件是否启用

首先确保你已经下载了Android NDK模块。然后在Android Studio中创建一个新应用，并将以下内容添加到ActivityMain文件中：

```
public class MainActivity implements Activity {
```

```

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}

Configure MyApp/app/build.gradle file

Edit the build.gradle file to look similar to the following example. Your version numbers may look different.

apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion 19
        buildToolsVersion "24.0.1"

        defaultConfig {
            applicationId "com.example.mydomain.myapp"
            minSdkVersion.apiLevel 19
            targetSdkVersion.apiLevel 19
            versionCode 1
            versionName "1.0"
        }
        buildTypes {
            release {
                minifyEnabled false
                proguardFiles.add(file('proguard-android.txt'))
            }
        }
    ndk {
        moduleName "myLib"

        /* The following lines are examples of some optional flags that
           you may set to configure your build environment
        */
        cppFlags.add("-I${file("path/to/my/includes/dir")}.toString())
        cppFlags.add("-std=c++11")
        ldLibs.addAll(['log', 'm'])
        stl = "c++_static"
        abiFilters.add("armeabi-v7a")
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}

```

Sync and check that there are no errors in the Gradle files before proceeding.

Test if plugin is enabled

First make sure you have downloaded the Android NDK module. Then create an new app in AndroidStudio and add the following to the ActivityMain file:

```
public class MainActivity implements Activity {
```

```

onCreate() {
    // 预生成代码。这里不重要
}
static {
    System.loadLibrary("myLib");
}
public static native String getString();
}

```

应将getString()部分高亮为红色，提示找不到对应的JNI函数。

将鼠标悬停在函数调用上，直到出现红色灯泡。点击灯泡并选择create function JNI...。这样会在myApp/app/src/main/jni目录下生成一个myLib.c文件，包含正确的JNI函数调用。它应类似如下：

```

#include <jni.h>

JNIEXPORT jstring JNICALL Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}

```

如果不是这样，说明插件未正确配置或NDK未下载。

第47.16节：显示签名信息

在某些情况下（例如获取Google API密钥），您需要找到您的密钥库指纹。Gradle有一个方便的任务可以显示所有签名信息，包括密钥库指纹：

```
./gradlew signingReport
```

这是一个示例输出：

```

:app:signingReport
变体：release
配置：无
-----
变体：debug
配置：debug
存储位置：/Users/user/.android/debug.keystore
别名：AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
有效期至：2044年6月18日星期六
-----
变体：debugAndroidTest
配置：debug
存储位置：/Users/user/.android/debug.keystore
别名：AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
有效期至：2044年6月18日星期六
-----
变体：debugUnitTest
配置：debug
存储位置：/Users/user/.android/debug.keystore
别名：AndroidDebugKey

```

```

onCreate() {
    // Pregenerated code. Not important here
}
static {
    System.loadLibrary("myLib");
}
public static native String getString();
}

```

The getString() part should be highlighted red saying that the corresponding JNI function could not be found. Hover your mouse over the function call until a red lightbulb appears. Click the bulb and select create function JNI.... This should generate a myLib.c file in the myApp/app/src/main/jni directory with the correct JNI function call. It should look similar to this:

```

#include <jni.h>

JNIEXPORT jstring JNICALL Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}

```

If it doesn't look like this, then the plugin has not correctly been configured or the NDK has not been downloaded

Section 47.16: Display signing information

In some circumstances (for example obtaining a Google API key) you need to find your keystore fingerprint. Gradle has a convenient task that display all the signing information, including keystore fingerprints:

```
./gradlew signingReport
```

This is a sample output:

```

:app:signingReport
Variant: release
Config: none
-----
Variant: debug
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
-----
Variant: debugAndroidTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
-----
Variant: debugUnitTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey

```

MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
有效期至：2044年6月18日星期六

变体：releaseUnitTest
配置：无

MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044

Variant: releaseUnitTest
Config: none

第47.17节：查看依赖树

使用任务依赖。根据你的模块设置，可能是 `./gradlew dependencies`
或者查看模块app的依赖使用 `./gradlew :app:dependencies`

以下是build.gradle文件示例

```
dependencies {
    compile 'com.android.support:design:23.2.1'
    compile 'com.android.support:cardview-v7:23.1.1'

    compile 'com.google.android.gms:play-services:6.5.87'
}
```

将生成以下图表：

并行执行是一个孵化中的功能。
`:app:dependencies`

项目 :app

```
...
_releaseApk - ## 内部使用，勿手动配置 ##
+--- com.android.support:design:23.2.1
|   +--- com.android.support:support-v4:23.2.1
|   |   \--- com.android.support:support-annotations:23.2.1
|   +--- com.android.support:appcompat-v7:23.2.1
|   |   +--- com.android.support:support-v4:23.2.1 (*)
|   |   +--- com.android.support:animated-vector-drawable:23.2.1
|   |   |   \--- com.android.support:support-vector-drawable:23.2.1 (*)
|   |   |       \--- com.android.support:support-v4:23.2.1 (*)
|   |   \--- com.android.support:support-vector-drawable:23.2.1 (*)
|   \--- com.android.support:recyclerview-v7:23.2.1
|       +--- com.android.support:support-v4:23.2.1 (*)
|       \--- com.android.support:support-annotations:23.2.1
+--- com.android.support:cardview-v7:23.1.1
\--- com.google.android.gms:play-services:6.5.87
\--- com.android.support:support-v4:21.0.0 -> 23.2.1 (*)
```

这里可以看到项目直接包含了`com.android.support:design`版本23.2.1，该版本自身引入了`com.android.support:support-v4`版本23.2.1。然而，`com.google.android.gms:play-services`本身依赖于相同的`support-v4`但版本较旧，为21.0.0，这就是gradle检测到的冲突。

(*) 表示gradle跳过了该子树，因为这些依赖之前已经列出过。

Section 47.17: Seeing dependency tree

Use the task dependencies. Depending on how your modules are set up, it may be either `./gradlew dependencies` or to see the dependencies of module app use `./gradlew :app:dependencies`

The example following build.gradle file

```
dependencies {
    compile 'com.android.support:design:23.2.1'
    compile 'com.android.support:cardview-v7:23.1.1'

    compile 'com.google.android.gms:play-services:6.5.87'
}
```

will produce the following graph:

Parallel execution is an incubating feature.
`:app:dependencies`

Project :app

```
...
_releaseApk - ## Internal use, do not manually configure ##
+--- com.android.support:design:23.2.1
|   +--- com.android.support:support-v4:23.2.1
|   |   \--- com.android.support:support-annotations:23.2.1
|   +--- com.android.support:appcompat-v7:23.2.1
|   |   +--- com.android.support:support-v4:23.2.1 (*)
|   |   +--- com.android.support:animated-vector-drawable:23.2.1
|   |   |   \--- com.android.support:support-vector-drawable:23.2.1 (*)
|   |   |       \--- com.android.support:support-v4:23.2.1 (*)
|   |   \--- com.android.support:support-vector-drawable:23.2.1 (*)
|   \--- com.android.support:recyclerview-v7:23.2.1
|       +--- com.android.support:support-v4:23.2.1 (*)
|       \--- com.android.support:support-annotations:23.2.1
+--- com.android.support:cardview-v7:23.1.1
\--- com.google.android.gms:play-services:6.5.87
\--- com.android.support:support-v4:21.0.0 -> 23.2.1 (*)
```

Here you can see the project is directly including `com.android.support:design` version 23.2.1, which itself is bringing `com.android.support:support-v4` with version 23.2.1. However, `com.google.android.gms:play-services` itself has a dependency on the same `support-v4` but with an older version 21.0.0, which is a conflict detected by gradle.

(*) are used when gradle skips the subtree because those dependencies were already listed previously.

第47.18节：禁用图像压缩以减小APK文件大小

如果您正在手动优化所有图片，请禁用APT Cruncher以获得更小的APK文件大小。

```
android {  
    aaptOptions {  
        cruncherEnabled = false  
    }  
}
```

第47.19节：自动删除“未对齐”apk

如果您不需要带有unaligned后缀的自动生成的apk文件（您很可能不需要），可以将以下代码添加到build.gradle文件中：

```
// 删除未对齐文件  
android.applicationVariants.all { variant ->  
    variant.assemble.doLast {  
        variant.outputs.each { output ->  
            println "aligned " + output.outputFile  
            println "unaligned " + output.packageApplication.outputFile  
  
            File unaligned = output.packageApplication.outputFile;  
            File aligned = output.outputFile  
            if (!unaligned.getName().equalsIgnoreCase(aligned.getName())) {  
                println "deleting " + unaligned.getName()  
                unaligned.delete()  
            }  
        }  
    }  
}
```

从这里开始

第47.20节：从gradle执行shell脚本

shell脚本是一种非常灵活的方式，可以将你的构建扩展到几乎任何你能想到的内容。

举个例子，这里有一个简单的脚本，用于编译protobuf文件并将生成的java文件添加到源代码目录以便进一步编译：

```
def compilePb() {  
    exec {  
        // 注意：如果protoc文件中有错误，gradle将会失败...  
        executable "../pbScript.sh"  
    }  
  
    project.afterEvaluate {  
        compilePb()  
    }
}
```

本例中的“pbScript.sh” shell脚本，位于项目根目录：

```
#!/usr/bin/env bash
```

Section 47.18: Disable image compression for a smaller APK file size

If you are optimizing all images manually, disable APT Cruncher for a smaller APK file size.

```
android {  
    aaptOptions {  
        cruncherEnabled = false  
    }  
}
```

Section 47.19: Delete "unaligned" apk automatically

If you don't need automatically generated apk files with unaligned suffix (which you probably don't), you may add the following code to build.gradle file:

```
// delete unaligned files  
android.applicationVariants.all { variant ->  
    variant.assemble.doLast {  
        variant.outputs.each { output ->  
            println "aligned " + output.outputFile  
            println "unaligned " + output.packageApplication.outputFile  
  
            File unaligned = output.packageApplication.outputFile;  
            File aligned = output.outputFile  
            if (!unaligned.getName().equalsIgnoreCase(aligned.getName())) {  
                println "deleting " + unaligned.getName()  
                unaligned.delete()  
            }  
        }  
    }  
}
```

From [here](#)

Section 47.20: Executing a shell script from gradle

A shell script is a very versatile way to extend your build to basically anything you can think of.

As an example, here is a simple script to compile protobuf files and add the result java files to the source directory for further compilation:

```
def compilePb() {  
    exec {  
        // NOTICE: gradle will fail if there's an error in the protoc file...  
        executable "../pbScript.sh"  
    }  
  
    project.afterEvaluate {  
        compilePb()  
    }
}
```

The 'pbScript.sh' shell script for this example, located in the project's root folder:

```
#!/usr/bin/env bash
```

```
pp=/home/myself/my/proto
```

```
/usr/local/bin/protoc -I=$pp \  
--java_out=./src/main/java \  
--proto_path=$pp \  
$pp/my.proto \  
--proto_path=$pp \  
$pp/my_other.proto
```

第47.21节：显示所有gradle项目任务

```
gradlew tasks -- 显示所有任务
```

Android任务

```
-----  
androidDependencies - 显示项目的Android依赖项。  
signingReport - 显示每个变体的签名信息。  
sourceSets - 打印出本项目中定义的所有源集。
```

构建任务

```
-----  
assemble - 组装所有应用程序和辅助包的所有变体。  
assembleAndroidTest - 组装所有测试应用程序。  
assembleDebug - 组装所有调试版本。  
assembleRelease - 组装所有发布版本。  
build - 组装并测试该项目。  
buildDependents - 组装并测试该项目及所有依赖该项目的项目。  
buildNeeded - 组装并测试该项目及所有该项目依赖的项目。
```

```
classes - 组装主类。  
clean - 删除构建目录。  
compileDebugAndroidTestSources
```

```
compileDebugSources  
compileDebugUnitTestSources  
compileReleaseSources
```

编译发布单元测试源代码

```
extractDebugAnnotations - 将调试版本的 Android 注释提取到归档文件中
```

```
extractReleaseAnnotations - 将发布版本的 Android
```

```
注释提取到归档文件中
```

```
jar - 组装包含主类的 jar 存档。
```

```
mockableAndroidJar - 创建一个适用于单元测试的 android.jar 版本。
```

```
testClasses - 组装测试类。
```

构建设置任务

```
-----  
init - 初始化一个新的 Gradle 构建。 [试验阶段]  
wrapper - 生成 Gradle 包装器文件。 [试验阶段]
```

文档任务

```
javadoc - 为主源代码生成 Javadoc API 文档。
```

帮助任务

```
-----  
buildEnvironment - 显示根项目 'LeitnerBoxPro' 中声明的所有构建脚本依赖项。  
components - 显示根项目 'LeitnerBoxPro' 生成的组件。 [试验中]  
dependencies - 显示根项目 'LeitnerBoxPro' 中声明的所有依赖项。  
dependencyInsight - 显示根项目
```

```
'LeitnerBoxPro' 中某个特定依赖项的详细信息。
```

```
help - 显示帮助信息。
```

```
pp=/home/myself/my/proto
```

```
/usr/local/bin/protoc -I=$pp \  
--java_out=./src/main/java \  
--proto_path=$pp \  
$pp/my.proto \  
--proto_path=$pp \  
$pp/my_other.proto
```

Section 47.21: Show all gradle project tasks

```
gradlew tasks -- show all tasks
```

Android tasks

```
-----  
androidDependencies - Displays the Android dependencies of the project.  
signingReport - Displays the signing info for each variant.  
sourceSets - Prints out all the source sets defined in this project.
```

Build tasks

```
-----  
assemble - Assembles all variants of all applications and secondary packages.  
assembleAndroidTest - Assembles all the Test applications.  
assembleDebug - Assembles all Debug builds.  
assembleRelease - Assembles all Release builds.  
build - Assembles and tests this project.  
buildDependents - Assembles and tests this project and all projects that depend on it.  
buildNeeded - Assembles and tests this project and all projects it depends on.  
classes - Assembles main classes.  
clean - Deletes the build directory.  
compileDebugAndroidTestSources  
compileDebugSources  
compileDebugUnitTestSources  
compileReleaseSources  
compileReleaseUnitTestSources  
extractDebugAnnotations - Extracts Android annotations for the debug variant into the archive file  
extractReleaseAnnotations - Extracts Android annotations for the release variant into the archive  
file  
jar - Assembles a jar archive containing the main classes.  
mockableAndroidJar - Creates a version of android.jar that is suitable for unit tests.  
testClasses - Assembles test classes.
```

Build Setup tasks

```
-----  
init - Initializes a new Gradle build. [incubating]  
wrapper - Generates Gradle wrapper files. [incubating]
```

Documentation tasks

```
-----  
javadoc - Generates Javadoc API documentation for the main source code.
```

Help tasks

```
-----  
buildEnvironment - Displays all buildscript dependencies declared in root project 'LeitnerBoxPro'.  
components - Displays the components produced by root project 'LeitnerBoxPro'. [incubating]  
dependencies - Displays all dependencies declared in root project 'LeitnerBoxPro'.  
dependencyInsight - Displays the insight into a specific dependency in root project  
'LeitnerBoxPro'.  
help - Displays a help message.
```

model - 显示根项目 'LeitnerBoxPro' 的配置模型。[试验中]
projects - 显示根项目 'LeitnerBoxPro' 的子项目。
properties - 显示根项目 'LeitnerBoxPro' 的属性。
tasks - 显示可从根项目 'LeitnerBoxPro' 运行的任务 (显示的部分任务可能属于子项目)
·

安装任务

installDebug - 安装调试版本。
installDebugAndroidTest - 安装调试版本的安卓(在设备上)测试。
uninstallAll - 卸载所有应用程序。
uninstallDebug - 卸载调试版本。
uninstallDebugAndroidTest - 卸载调试版本的安卓(设备上)测试。
uninstallRelease - 卸载发布版本。

Verification 任务

check - 运行所有检查。
connectedAndroidTest - 安装并运行所有风味的仪器测试于连接的设备上。
connectedCheck - 在当前连接的设备上运行所有设备检查。
connectedDebugAndroidTest - 安装并运行调试版本的测试于连接的设备上。
deviceAndroidTest - 使用所有设备提供者安装并运行仪器测试。
deviceCheck - 使用设备提供者和测试服务器运行所有设备检查。
lint - 对所有变体运行lint检查。
lintDebug - 在调试版本上运行lint。
lintRelease - 在发布版本上运行lint。
test - 运行所有变体的单元测试。
testDebugUnitTest - 运行调试版本的单元测试。
testReleaseUnitTest - 运行发布版本的单元测试。

其他任务

assembleDefault
clean
jarDebugClasses
jarReleaseClasses
transformResourcesWithMergeJavaResForDebugUnitTest
transformResourcesWithMergeJavaResForReleaseUnitTest

model - Displays the configuration model of root project 'LeitnerBoxPro'. [incubating]
projects - Displays the sub-projects of root project 'LeitnerBoxPro'.
properties - Displays the properties of root project 'LeitnerBoxPro'.
tasks - Displays the tasks runnable from root project 'LeitnerBoxPro' (some of the displayed tasks may belong to subprojects)
·

Install tasks

installDebug - Installs the Debug build.
installDebugAndroidTest - Installs the android (on device) tests for the Debug build.
uninstallAll - Uninstall all applications.
uninstallDebug - Uninstalls the Debug build.
uninstallDebugAndroidTest - Uninstalls the android (on device) tests for the Debug build.
uninstallRelease - Uninstalls the Release build.

Verification tasks

check - Runs all checks.
connectedAndroidTest - Installs and runs instrumentation tests for all flavors on connected devices.
connectedCheck - Runs all device checks on currently connected devices.
connectedDebugAndroidTest - Installs and runs the tests for debug on connected devices.
deviceAndroidTest - Installs and runs instrumentation tests using all Device Providers.
deviceCheck - Runs all device checks using Device Providers and Test Servers.
lint - Runs lint on all variants.
lintDebug - Runs lint on the Debug build.
lintRelease - Runs lint on the Release build.
test - Run unit tests for all variants.
testDebugUnitTest - Run unit tests for the debug build.
testReleaseUnitTest - Run unit tests for the release build.

Other tasks

assembleDefault
clean
jarDebugClasses
jarReleaseClasses
transformResourcesWithMergeJavaResForDebugUnitTest
transformResourcesWithMergeJavaResForReleaseUnitTest

第47.22节：调试你的Gradle错误

以下内容摘自Gradle - 什么是非零退出值以及如何修复？，完整讨论请参见原文。

假设你正在开发一个应用程序，遇到了一些Gradle错误，通常看起来像这样

```
:模块:某任务 失败
失败: 构建因异常失败。
* 出了什么问题:
任务':module:someTask'执行失败。
> 这里有一些信息... 以非零退出值X结束
* 尝试:
使用--stacktrace选项运行以获取堆栈跟踪。使用--info或--debug选项运行以获取更多日志输出。
构建失败
总时间: Y.ZZ秒
```

Section 47.22: Debugging your Gradle errors

The following is an excerpt from [Gradle - What is a non-zero exit value and how do I fix it?](#), see it for the full discussion.

Let's say you are developing an application and you get some Gradle error that appears that generally will look like so.

```
:module:someTask FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':module:someTask'.
> some message here... finished with non-zero exit value X
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
BUILD FAILED
Total time: Y.ZZ secs
```

你在StackOverflow上搜索你的问题，人们建议清理并重建项目，或启用[MultiDex](#)，但当你尝试这些方法时，问题依然没有解决。

有方法可以获取更多信息，但Gradle输出本身应该指向该消息上方几行中实际的错误，位于 :module:someTask FAILED 和最后一个通过的 :module:someOtherTask 之间。

因此，如果你要询问关于错误的问题，请编辑你的问题以包含更多错误的上下文。

所以，你得到了一个“非零退出值”。这个数字是你应该尝试修复的一个很好的指标。以下是一些最常见的情况。

- 1是一个通用错误代码，错误很可能出现在Gradle输出中
- 2似乎与依赖重叠或项目配置错误有关。
- 3似乎是由于包含过多依赖项或内存问题引起的。

上述问题的一般解决方案（在尝试清理并重建项目之后）是：

- 1- 解决提到的错误。通常，这是一个编译时错误，意味着项目中的某段代码无效。这包括Android项目中的XML和Java代码。
- 2 & 3 - 这里的许多答案告诉你启用[multidex](#)。虽然这可能解决问题，但更可能是一个解决方法。如果你不明白为什么要使用它（请参见链接），你可能根本不需要它。一般的解决方案包括减少对库依赖的过度使用（例如，当你只需要使用一个库，比如地图或登录时，却引入了整个谷歌服务库）。

第47.23节：使用gradle.properties进行中央版本号/构建配置管理

您可以在以下位置定义中央配置信息

- 一个单独的gradle包含文件 通过“dependencies.gradle”文件集中管理依赖
- 一个独立的属性文件 通过“version.properties”文件进行构建版本控制

或者使用根目录的gradle.properties文件

项目结构

```
根目录
+- module1/
|   build.gradle
+- module2/
|   build.gradle
+- build.gradle
+- gradle.properties
```

gradle.properties中所有子模块的全局设置

```
# 用于manifest
# todo 每次发布递增
appVersionCode=19
appVersionName=0.5.2.160726

# android工具设置
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

子模块中的使用

You search here on StackOverflow for your problem, and people say to clean and rebuild your project, or enable [MultiDex](#), and when you try that, it just isn't fixing the problem.

[There are ways to get more information](#), but the Gradle output itself should point at the actual error in the few lines above that message between :module:someTask FAILED and the last :module:someOtherTask that passed. Therefore, if you ask a question about your error, please edit your questions to include more context to the error.

So, you get a "non-zero exit value." Well, that number is a good indicator of what you should try to fix. Here are a few occur most frequently.

- 1 is a just a general error code and the error is likely in the Gradle output
- 2 seems to be related to overlapping dependencies or project misconfiguration.
- 3 seems to be from including too many dependencies, or a memory issue.

The general solutions for the above (after attempting a Clean and Rebuild of the project) are:

- 1 - Address the error that is mentioned. Generally, this is a compile-time error, meaning some piece of code in your project is not valid. This includes both XML and Java for an Android project.
- 2 & 3 - Many answers here tell you to enable [multidex](#). While it may fix the problem, it is most likely a workaround. If you don't understand why you are using it (see the link), you probably don't need it. General solutions involve cutting back your overuse of library dependencies (such as all of Google Play Services, when you only need to use one library, like Maps or Sign-In, for example).

Section 47.23: Use gradle.properties for central versionnumber/buildconfigurations

You can define central config info's in

- a separate gradle include file Centralizing dependencies via "dependencies.gradle" file
- a stand alone properties file Versioning your builds via "version.properties" file

or do it with root gradle.properties file

the project structure

```
root
+- module1/
|   build.gradle
+- module2/
|   build.gradle
+- build.gradle
+- gradle.properties
```

global setting for all submodules in gradle.properties

```
# used for manifest
# todo increment for every release
appVersionCode=19
appVersionName=0.5.2.160726

# android tools settings
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

usage in a submodule

```

apply plugin: 'com.android.application'
android {
    // appXXX 在 gradle.properties 中定义
    compileSdkVersion = Integer.valueOf(appCompileSdkVersion)
    buildToolsVersion = appBuildToolsVersion

    defaultConfig {
        // appXXX 在 gradle.properties 中定义
        versionCode = Long.valueOf(appVersionCode)
        versionName = appVersionName
    }
}

dependencies {
    ...
}

```

注意：如果你想在 F-Droid 应用商店发布你的应用，必须在 gradle 文件中使用魔术数字
否则 f-droid 机器人无法读取当前版本号来检测/验证版本变更。

第 47.24 节：定义构建类型

你可以在模块级别的 build.gradle 文件中的 android {} 块内创建和配置构建类型。

```

android {
    ...
    defaultConfig {...}

    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }

        debug {
            applicationIdSuffix ".debug"
        }
    }
}

```

```

apply plugin: 'com.android.application'
android {
    // appXXX are defined in gradle.properties
    compileSdkVersion = Integer.valueOf(appCompileSdkVersion)
    buildToolsVersion = appBuildToolsVersion

    defaultConfig {
        // appXXX are defined in gradle.properties
        versionCode = Long.valueOf(appVersionCode)
        versionName = appVersionName
    }
}

dependencies {
    ...
}

```

Note: If you want to publish your app in the F-Droid app store you have to use magic numbers in the gradle file because else f-droid robot cannot read current versionnumber to detect/verify version changes.

Section 47.24: Defining build types

You can create and configure build types in the module-level build.gradle file inside the android {} block.

```

android {
    ...
    defaultConfig {...}

    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }

        debug {
            applicationIdSuffix ".debug"
        }
    }
}

```

第48章：Android 文件输入输出

在 Android 中读写文件与在标准 Java 中读写文件没有区别。可以使用相同的java.io包。然而，有一些特定的内容涉及你被允许写入的文件夹、权限管理以及 MTP 的解决方法。

第48.1节：获取工作文件夹

你可以通过在你的 Activity 上调用方法getFilesDir()来获取工作文件夹（Activity 是你应用中的核心类，继承自 Context 。详见此处）。读取操作没有区别。只有你的应用可以访问该文件夹。

例如，你的 Activity 可能包含以下代码：

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

第48.2节：写入原始字节数组

```
File myFile = new File(getFilesDir(), "myData.bin");
FileOutputStream out = new FileOutputStream(myFile);

// 写入四个字节 — 二 三 四：
out.write(new byte[] { 1, 2, 3, 4 })
out.close()
```

这段代码没有任何 Android 特有的内容。如果你经常写入大量小数据，使用BufferedOutputStream可以减少设备内部 SSD 的磨损。

第48.3节：对象序列化

Android 支持传统的 Java 对象序列化。你可以定义实现 Serializable 接口的类，例如：

```
class Circle implements Serializable {
    final int radius;
    final String name;

    Circle(int radius, int name) {
        this.radius = radius;
        this.name = name;
    }
}
```

然后将它们写入 ObjectOutputStream :

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

Java对象序列化可能是完美的选择，也可能非常糟糕的选择，这取决于你想用它做什么——

Chapter 48: FileIO with Android

Reading and writing files in Android are not different from reading and writing files in standard Java. Same java.io package can be used. However, there is some specific related to the folders where you are allowed to write, permissions in general and MTP work arounds.

Section 48.1: Obtaining the working folder

You can get your working folder by calling the method [getFilesDir\(\)](#) on your Activity (Activity is the central class in your application that inherits from Context. See here). Reading is not different. Only your application will have access to this folder.

Your activity could contain the following code, for instance:

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

Section 48.2: Writing raw array of bytes

```
File myFile = new File(getFilesDir(), "myData.bin");
FileOutputStream out = new FileOutputStream(myFile);

// Write four bytes one two three four:
out.write(new byte[] { 1, 2, 3, 4 })
out.close()
```

There is nothing Android specific with this code. If you write lots of small values often, use [BufferedOutputStream](#) to reduce the wear of the device internal SSD.

Section 48.3: Serializing the object

The old good Java object serialization is available for you in Android. you can define Serializable classes like:

```
class Circle implements Serializable {
    final int radius;
    final String name;

    Circle(int radius, int name) {
        this.radius = radius;
        this.name = name;
    }
}
```

and then write then to the ObjectOutputStream:

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

Java object serialization may be either perfect or really bad choice, depending on what do you want to do with it -

第48.4节：写入外部存储 (SD卡)

你也可以从许多安卓设备中存在的内存卡 (SD卡) 读取和写入文件。该位置的文件可以被其他程序访问，用户在通过USB线连接设备到电脑并启用MTP协议后，也可以直接访问这些文件。

找到SD卡的位置有些问题。Environment类包含静态方法来获取“外部目录”，这些目录通常位于SD卡内，还能获取SD卡是否存在以及是否可写的信息。这个问题包含了如何确保找到正确位置的宝贵答案。

访问外部存储需要在你的Android清单文件中声明权限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

对于较旧版本的Android，只需将权限写入清单文件 (manifest) 中（用户必须在安装时批准）。但从Android 6.0开始，Android会在首次访问时请求用户批准，您必须支持这种新方式。否则，无论清单文件中如何设置，访问都会被拒绝。

在Android 6.0中，首先需要检查权限，如果未授予，则请求权限。代码示例可以在[this SO question](#)中找到。

第48.5节：解决“不可见的MTP文件”问题

如果您通过USB线使用MTP协议创建文件以导出到桌面，可能会遇到新创建的文件在连接的桌面PC上的文件管理器中无法立即显示的问题。要使新文件可见，您需要调用MediaScannerConnection：

```
File file = new File(Environment.getExternalStoragePublicDirectory(
    Environment.DIRECTORY_DOCUMENTS), "theDocument.txt");
FileOutputStream out = new FileOutputStream(file)

... (write the document)

out.close()
MediaScannerConnection.scanFile(this, new String[] {file.getPath()}, null, null);
context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
Uri.fromFile(file)));
```

此MediaScannerConnection调用代码仅适用于文件，不适用于目录。该问题在[this Android bug report](#)中有描述。未来某些版本或某些设备上可能会修复此问题。

第48.6节：处理大文件

小文件可以在几分之一秒内处理完毕，您可以在需要的代码位置直接读写它们。但是如果文件较大或处理速度较慢，您可能需要在Android中使用AsyncTask来后台处理文件：

```
class FileOperation extends AsyncTask<String, Void, File> {

    @Override
```

outside the scope of this tutorial and sometimes opinion based. Read about the [versioning](#) first if you decide to use it.

Section 48.4: Writing to external storage (SD card)

You can also read and write from/to memory card (SD card) that is present in many Android devices. Files in this location can be accessed by other programs, also directly by the user after connecting device to PC via USB cable and enabling MTP protocol.

Finding the SD card location is somewhat more problematic. The [Environment](#) class contains static methods to get "external directories" that should normally be inside the SD card, also information if the SD card exists at all and is writable. [This question](#) contains valuable answers how to make sure the right location will be found.

Accessing external storage requires permissions in your Android manifest:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

For older versions of Android putting permissions in manifest (the user must approve during installation). However starting from Android 6.0 Android asks the user for approval at the time of the first access, and you must support this new approach. Otherwise access is denied regardless of your manifest.

In Android 6.0, first you need to check for permission, then, if not granted, request it. The code examples can be found inside [this SO question](#).

Section 48.5: Solving "Invisible MTP files" problem

If you create files for exporting via USB cable to desktop using MTP protocol, may be a problem that newly created files are not immediately visible in the file explorer running on the connected desktop PC. To make new files visible, you need to call [MediaScannerConnection](#):

```
File file = new File(Environment.getExternalStoragePublicDirectory(
    Environment.DIRECTORY_DOCUMENTS), "theDocument.txt");
FileOutputStream out = new FileOutputStream(file)

... (write the document)

out.close()
MediaScannerConnection.scanFile(this, new String[] {file.getPath()}, null, null);
context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
Uri.fromFile(file)));
```

This MediaScannerConnection call code works for files only, not for directories. The problem is described in [this Android bug report](#). This may be fixed for some version in the future, or on some devices.

Section 48.6: Working with big files

Small files are processed in a fraction of second and you can read / write them in place of the code where you need this. However if the file is bigger or otherwise slower to process, you may need to use AsyncTask in Android to work with the file in the background:

```
class FileOperation extends AsyncTask<String, Void, File> {

    @Override
```

```

protected File doInBackground(String... params) {
    try {
        File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_DOCUMENTS), "bigAndComplexDocument.odf");
        FileOutputStream out = new FileOutputStream(file)

        ... (写入文档)

        out.close()
        return file;
    } catch (IOException ex) {
        Log.e("无法写入", ex);
        return null;
    }
}

@Override
protected void onPostExecute(File result) {
    // 这是在完成时调用的
}

@Override
protected void onPreExecute() {
    // 这是在开始之前调用的
}

@Override
protected void onProgressUpdate(Void... values) {
    // 这个例子中不太可能需要
}
}

```

然后

```
new FileOperation().execute("Some parameters");
```

这个 SO 问题 包含了如何创建和调用 AsyncTask 的完整示例。另请参见关于错误 处理，关于如何处理 IOExceptions 和其他错误。

```

protected File doInBackground(String... params) {
    try {
        File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_DOCUMENTS), "bigAndComplexDocument.odf");
        FileOutputStream out = new FileOutputStream(file)

        ... (write the document)

        out.close()
        return file;
    } catch (IOException ex) {
        Log.e("Unable to write", ex);
        return null;
    }
}

@Override
protected void onPostExecute(File result) {
    // This is called when we finish
}

@Override
protected void onPreExecute() {
    // This is called before we begin
}

@Override
protected void onProgressUpdate(Void... values) {
    // Unlikely required for this example
}
}

```

and then

```
new FileOperation().execute("Some parameters");
```

[This SO question](#) contains the complete example on how to create and call the AsyncTask. Also see the [question on error](#) handling on how to handle IOExceptions and other errors.

第49章：FileProvider

第49.1节：共享文件

在本例中，您将学习如何与其他应用共享文件。虽然代码适用于所有其他格式，但本例中我们将使用一个 PDF 文件。

路线图：

指定放置要共享文件的目录

为了共享文件，我们将使用 FileProvider，这是一个允许应用间安全共享文件的类。FileProvider 只能共享预定义目录中的文件，因此我们需要定义这些目录。

1. 创建一个新的 XML 文件来包含路径，例如res/xml/filepaths.xml
2. 添加路径

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/" />
</paths>
```

定义一个 FileProvider 并将其与文件路径关联

这在清单文件中完成：

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    ...
</manifest>
```

生成文件的URI

要共享文件，我们必须为文件提供一个标识符。这是通过使用URI（统一资源标识符）来完成的。

```
// 我们假设要加载的文件位于内部存储的documents/子目录中

File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// 当然，这也行：
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

如代码所示，我们首先创建了一个表示该文件的File类。要获取URI，我们请求FileProvider为我们获取一个。第二个参数很重要：它传递了FileProvider的权限（authority）。它必须与manifest中定义的FileProvider的权限相同。

Chapter 49: FileProvider

Section 49.1: Sharing a file

In this example you'll learn how to share a file with other apps. We'll use a pdf file in this example although the code works with every other format as well.

The roadmap:

Specify the directories in which the files you want to share are placed

To share files we'll use a FileProvider, a class allowing secure file sharing between apps. A FileProvider can only share files in predefined directories, so let's define these.

1. Create a new XML file that will contain the paths, e.g. res/xml/filepaths.xml
2. Add the paths

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/" />
</paths>
```

Define a FileProvider and link it with the file paths

This is done in the manifest:

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
    ...
</manifest>
```

Generate the URI for the file

To share the file we must provide an identifier for the file. This is done by using a URI (Uniform Resource Identifier).

```
// We assume the file we want to load is in the documents/ subdirectory
// of the internal storage
File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// This can also in one line of course:
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

As you can see in the code we first make a new File class representing the file. To get a URI we ask FileProvider to get us one. The second argument is important: it passes the authority of a FileProvider. It must be equal to the authority of the FileProvider defined in the manifest.

与其他应用共享文件

我们使用 ShareCompat 与其他应用共享文件：

```
Intent intent = ShareCompat.IntentBuilder.from(getContext())
    .setType("application/pdf")
.setStream(uri)
    .setChooserTitle("选择条形码")
.createChooserIntent()
.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

Context.startActivity(intent);
```

选择器是一个菜单，用户可以从中选择想要用来共享文件的应用。标志

Intent.FLAG_GRANT_READ_URI_PERMISSION 用于授予对 URI 的临时读取权限。

Share the file with other apps

We use ShareCompat to share the file with other apps:

```
Intent intent = ShareCompat.IntentBuilder.from(getContext())
    .setType("application/pdf")
.setStream(uri)
    .setChooserTitle("Choose bar")
.createChooserIntent()
.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

Context.startActivity(intent);
```

A chooser is a menu from which the user can choose with which app he/she wants to share the file. The flag

Intent.FLAG_GRANT_READ_URI_PERMISSION is needed to grant temporary read access permission to the URI.

第50章：在内部和外部存储中存储文件

参数	详细信息
姓名	要打开的文件名。注意：不能包含路径分隔符
模式	操作模式。默认操作使用MODE_PRIVATE，追加到现有文件使用MODE_APPEND。 其他模式包括MODE_WORLD_READABLE和MODE_WORLD_WRITEABLE，这两者在API 17中已被弃用。
目录	创建新文件的目录
路径	指定新文件位置的路径
类型	要检索的文件目录类型。可以是null，或以下任意一种：DIRECTORY_MUSIC、 DIRECTORY_PODCASTS、DIRECTORY_RINGTONES、DIRECTORY_ALARMS、DIRECTORY_NOTIFICATIONS、 DIRECTORY_PICTURES或DIRECTORY_MOVIES

第50.1节：Android：内部存储和外部存储 - 术语澄清

Android开发者（主要是初学者）对内部存储和外部存储的术语感到困惑。

在Stackoverflow上有很多关于此问题的提问。这主要是因为根据Google/官方Android文档的术语与普通Android操作系统用户的理解有很大不同。因此我认为记录下来会有所帮助。

我们所理解的 - 用户的术语 (UT)

内部存储 (UT)	外部存储 (UT)
手机的内置内部存储	可拆卸的安全数字 (SD) 卡或micro SD存储
示例：Nexus 6P的32 GB内部存储。	示例：由三星、闪迪、Strontium、创见等厂商提供的可拆卸SD卡中的存储空间

但是，根据Android文档/指南 - Google的术语 (GT)

内部存储 (GT)：

默认情况下，保存到内部存储的文件对您的应用程序是私有的，其他应用程序无法访问它们（用户也无法访问）。

外部存储 (GT)：

这可以是可移动存储介质（例如SD卡）或内部（不可移动）存储。

外部存储 (GT) 可以分为两种类型：

主外部存储	次外部存储或可移动存储 (GT)
这与手机内置的内部存储（或）内部存储 (UT) 相同	这与可移动的micro SD卡存储（或）外部存储 (UT) 相同
示例：Nexus 6P的32 GB内部存储。	示例：由三星、闪迪、Strontium、创见等供应商提供的可移动SD卡中的存储空间

Chapter 50: Storing Files in Internal & External Storage

Parameter	Details
name	The name of the file to open. NOTE: Cannot contain path separators
mode	Operating mode. Use MODE_PRIVATE for default operation, and MODE_APPEND to append an existing file. Other modes include MODE_WORLD_READABLE and MODE_WORLD_WRITEABLE, which were both deprecated in API 17.
dir	Directory of the file to create a new file in
path	Path to specify the location of the new file
type	Type of files directory to retrieve. Can be null, or any of the following: DIRECTORY_MUSIC, DIRECTORY_PODCASTS, DIRECTORY_RINGTONES, DIRECTORY_ALARMS, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES, or DIRECTORY_MOVIES

Section 50.1: Android: Internal and External Storage - Terminology Clarification

Android developers(mainly beginners) have been confused regarding Internal & External storage terminology. There are lot of questions on Stackoverflow regarding the same. This is mainly because of the fact that terminology according to Google/official Android documentation is quite different to that of normal Android OS user. Hence I thought documenting this would help.

What we think - User's Terminology (UT)

Internal storage(UT)	External storage(UT)
phone's inbuilt internal memory	removable Secure Digital(SD) card or micro SD storage
Example: Nexus 6P's 32 GB internal memory.	Example: storage space in removable SD cards provided by vendors like samsung, sandisk, strontium, transcend and others

But, According to Android Documentation/Guide - Google's Terminology (GT)

Internal storage(GT):

By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user).

External storage(GT):

This can be a removable storage media (such as an SD card) or an internal (non-removable) storage.

External Storage(GT) can be categorized into two types:

Primary External Storage	Secondary External Storage or Removable storage(GT)
This is same as phone's inbuilt internal memory (or) Internal storage(UT)	This is same as removable micro SD card storage (or) External storage(UT)
Example: Nexus 6P's 32 GB internal memory.	Example: storage space in removable SD cards provided by vendors like samsung, sandisk, strontium, transcend and others

这种存储类型可以通过将手机通过USB线连接到Windows电脑，并在USB选项通知中选择Camera(PTP)来访问。

这种存储类型可以通过将手机通过USB线连接到Windows电脑，并在USB选项通知中选择Filetransfer来访问。

简而言之，

外部存储(GT) = 内部存储(UT) 和 外部存储(UT)

可移动存储(GT) = 外部存储(UT)

内部存储(GT)在UT中没有对应的术语。

让我来详细说明，

内部存储(GT)：默认情况下，保存到内部存储的文件是应用私有的，其他应用无法访问这些文件。即使用户在文件管理器中启用了“显示隐藏文件”选项，用户也无法通过文件管理器访问这些文件。要访问内部存储(GT)中的文件，必须对Android手机进行root。此外，当用户卸载应用时，这些文件会被删除。

所以内部存储(GT) 不是 我们通常理解的Nexus 6P的32/64 GB内部存储

一般来说，**内部存储(GT)的位置通常是：**

/data/data/你的.应用.包名/某个目录/

外部存储(GT)：

每个兼容安卓的设备都支持一个共享的“外部存储”，你可以用它来保存文件。保存到外部存储的文件是全局可读的，当用户启用USB大容量存储以在电脑上传输文件时，这些文件也可以被用户修改。

外部存储 (GT) 位置：它可以位于您的内部存储 (UT) 中的任何地方，或者位于您的可移动存储 (GT) 中即 micro SD 卡。具体位置取决于您的手机制造商 (OEM) 以及 Android 操作系统版本。

为了在外部存储 (GT) 上读取或写入文件，您的应用必须获取READ_EXTERNAL_STORAGE或WRITE_EXTERNAL_STORAGE系统权限。

例如：

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

如果您需要同时读取和写入文件，则只需请求WRITE_EXTERNAL_STORAGE权限，因为它隐含地也需要读取权限。

在外部存储 (GT) 中，您也可以保存应用私有

的文件，但

当用户卸载您的应用时，该目录及其所有内容都会被删除。

This type of storage can be accessed on windows PC by connecting your phone to PC via USB cable and selecting Camera(PTP) in the USB options notification.

This type of storage can be accessed on windows PC by connecting your phone to PC via USB cable and selecting File transfer in the USB options notification.

In a nutshell,

External Storage(GT) = Internal Storage(UT) and External Storage(UT)

Removable Storage(GT) = External Storage(UT)

Internal Storage(GT) doesn't have a term in UT.

Let me explain clearly,

Internal Storage(GT): By default, files saved to the internal storage are private to your application and other applications cannot access them. Your app user also can't access them using file manager; even after enabling "show hidden files" option in file manager. To access files in Internal Storage(GT), you have to root your Android phone. Moreover, when the user uninstalls your application, these files are removed/deleted.

So Internal Storage(GT) is **NOT** what we think as Nexus 6P's 32/64 GB internal memory

Generally, **Internal Storage(GT) location** would be:

/data/data/your.application.package.appname/someDirectory/

External Storage(GT):

Every Android-compatible device supports a shared "external storage" that you can use to save files. Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

External Storage(GT) location: It could be *anywhere* in your internal storage(UT) or in your removable storage(GT) i.e. micro SD card. It depends on your phone's OEM and also on Android OS version.

In order to read or write files on the External Storage(GT), your app must acquire the READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE system permissions.

For example:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

If you need to both read and write files, then you need to request only the WRITE_EXTERNAL_STORAGE permission, because it implicitly requires read access as well.

In **External Storage(GT)**, you may also save files that are **app-private**

But,

When the user uninstalls your application, this directory and all its contents are deleted.

什么时候需要在外部存储 (GT) 中保存应用私有文件？

如果您处理的是不打算供其他应用使用的文件（例如仅由您的应用使用的图形纹理或音效），则应使用外部存储上的私有存储目录

从 Android 4.4 开始，读取或写入应用私有目录中的文件不再需要 READ_EXTERNAL_STORAGE 或 WRITE_EXTERNAL_STORAGE 权限。因此，您可以通过添加 maxSdkVersion 属性，仅在较低版本的 Android 上声明需要请求该权限：

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="18" />
    ...
</manifest>
```

存储到内部存储的方法 (GT)：

这两种方法都存在于 Context 类中

```
File getDir (String name, int mode)
File getFilesDir ()
```

存储到主外部存储 (即内部存储) (UT) 的方法：

```
File getExternalStorageDirectory ()
File getExternalFilesDir (String type)
File getExternalStoragePublicDirectory (String type)
```

一开始，大家都使用 Environment.getExternalStorageDirectory()，它指向 Primary 的根目录外部存储。因此，主外部存储被填充了随机内容。

后来，添加了以下两种方法：

1. 在 Context 类中，添加了 getExternalFilesDir()，指向主外部存储上的应用专用目录。
当应用被卸载时，该目录及其内容将被删除。
2. Environment.getExternalStoragePublicDirectory() 用于集中存放常见文件类型，如照片和视频。该目录及其内容在应用卸载时不会被删除。

存储到可移动存储 (GT)，即 micro SD 卡的方法

在 API 等级 19 之前，没有官方方法可以存储到 SD 卡。但许多人可以使用非官方库或 API 实现。

官方方法是在 API 等级 19 (Android 版本 4.4 - Kitkat) 的 Context 类中引入的。

```
File[] getExternalFilesDirs (String type)
```

When do you need to save files that are app-private in External Storage(GT)?

If you are handling files that are not intended for other apps to use (such as graphic textures or sound effects used by only your app), you should use a private storage directory on the external storage

Beginning with Android 4.4, reading or writing files in your app's private directories does not require the READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE permissions. So you can declare the permission should be requested only on the lower versions of Android by adding the maxSdkVersion attribute:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="18" />
    ...
</manifest>
```

Methods to store in Internal Storage(GT):

Both these methods are present in [Context](#) class

```
File getDir (String name, int mode)
File getFilesDir ()
```

Methods to store in Primary External Storage i.e. Internal Storage(UT):

```
File getExternalStorageDirectory ()
File getExternalFilesDir (String type)
File getExternalStoragePublicDirectory (String type)
```

In the beginning, everyone used [Environment.getExternalStorageDirectory\(\)](#)，which pointed to the **root of Primary External Storage**. As a result, Primary External Storage was filled with random content.

Later, these two methods were added:

1. In [Context](#) class, they added [getExternalFilesDir\(\)](#)，指向一个 **app-specific directory** on Primary External Storage. This directory and its contents **will be deleted** when the app is uninstalled.
2. [Environment.getExternalStoragePublicDirectory\(\)](#) for centralized places to store well-known file types, like photos and movies. This directory and its contents **will NOT be deleted** when the app is uninstalled.

Methods to store in Removable Storage(GT) i.e. micro SD card

Before **API level 19**, there was **no official way** to store in SD card. But, many could do it using unofficial libraries or APIs.

Officially, one method was introduced in [Context](#) class in API level 19 (Android version 4.4 - Kitkat).

```
File[] getExternalFilesDirs (String type)
```

它返回应用程序专用目录在所有共享/外部存储设备上的绝对路径，应用程序可以在这些目录中放置其拥有的持久文件。这些文件是应用程序内部文件，通常不会作为媒体文件被用户看到。

It returns absolute paths to application-specific directories on all shared/external storage devices where the application can place persistent files it owns. These files are internal to the application, and not typically visible to the user as media.

这意味着，它将返回两种类型的外部存储 (GT) 路径——内部存储和Micro SD卡。

通常第二个路径会是Micro SD卡的存储路径（但不总是如此）。因此你需要通过执行带有此方法的代码来检查。

示例代码片段：

我创建了一个带空活动的新安卓项目，在其中写入了以下代码

在MainActivity.java的protected void onCreate(Bundle savedInstanceState)方法中

```
File internal_m1 = getDir("custom", 0);
File internal_m2 = getFilesDir();

File external_m1 = Environment.getExternalStorageDirectory();

File external_m2 = getExternalFilesDir(null);
File external_m2_Args = getExternalFilesDir(Environment.DIRECTORY_PICTURES);

File external_m3 =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

File[] external_AND_removable_storage_m1 = getExternalFilesDirs(null);
File[] external_AND_removable_storage_m1_Args =
getExternalFilesDirs(Environment.DIRECTORY_PICTURES);
```

执行上述代码后，

输出：

```
internal_m1: /data/data/your.application.package.appname/app_custom
internal_m2: /data/data/your.application.package.appname/files
external_m1: /storage/emulated/0
external_m2: /storage/emulated/0/Android/data/your.application.package.appname/files
external_m2_Args: /storage/emulated/0/Android/data/your.application.package.appname/files/Pictures
external_m3: /storage/emulated/0/Pictures
external_AND_removable_storage_m1 (第一个路径):
/storage/emulated/0/Android/data/your.application.package.appname/files
external_AND_removable_storage_m1 (第二个路径):
/storage/sdcard1/Android/data/your.application.package.appname/files
external_AND_removable_storage_m1_Args (第一个路径):
/storage/emulated/0/Android/data/your.application.package.appname/files/Pictures
external_AND_removable_storage_m1_Args (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files/Pictures
```

注意：我已将手机连接到Windows电脑；启用了开发者选项和USB调试，然后运行了

That means, it will return paths to **both** types of External Storage(GT) - Internal memory and Micro SD card. Generally **second path** would be storage path of micro SD card(but not always). So you need to check it out by executing the code with this method.

Example with code snippet:

I created a new android project with empty activity, wrote the following code inside

protected void onCreate(Bundle savedInstanceState) method of MainActivity.java

```
File internal_m1 = getDir("custom", 0);
File internal_m2 = getFilesDir();

File external_m1 = Environment.getExternalStorageDirectory();

File external_m2 = getExternalFilesDir(null);
File external_m2_Args = getExternalFilesDir(Environment.DIRECTORY_PICTURES);

File external_m3 =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

File[] external_AND_removable_storage_m1 = getExternalFilesDirs(null);
File[] external_AND_removable_storage_m1_Args =
getExternalFilesDirs(Environment.DIRECTORY_PICTURES);
```

After executing above code,

Output:

```
internal_m1: /data/data/your.application.package.appname/app_custom
internal_m2: /data/data/your.application.package.appname/files
external_m1: /storage/emulated/0
external_m2: /storage/emulated/0/Android/data/your.application.package.appname/files
external_m2_Args: /storage/emulated/0/Android/data/your.application.package.appname/files/Pictures
external_m3: /storage/emulated/0/Pictures
external_AND_removable_storage_m1 (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files
external_AND_removable_storage_m1 (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files
external_AND_removable_storage_m1_Args (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files/Pictures
external_AND_removable_storage_m1_Args (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files/Pictures
```

Note: I have connected my phone to Windows PC; enabled both developer options, USB debugging and then ran

这段代码。如果你**没有连接手机**；而是在**Android模拟器**上运行，输出结果可能会有所不同。我的手机型号是Coolpad Note 3，运行Android 5.1系统

我手机上的存储位置：

Micro SD存储位置: /storage/sdcard1

内部存储 (UT) 位置: /storage/sdcard0。

注意/sdcard 和 /storage/emulated/0 也指向内部存储 (UT)。但它们是指向/storage/sdcard0的符号链接。

要清楚了解Android中不同的存储路径，请查看这个答案

免责声明：上述所有存储路径均为我手机上的路径。你的文件可能不存储在相同的存储路径上。因为存储位置/路径可能因手机厂商、制造商及不同版本的Android操作系统而异。

第50.2节：使用外部存储

“外部”存储是我们可以用来将文件保存到用户设备的另一种存储类型。它与“内部”存储有一些关键的区别，具体如下：

- 它并不总是可用的。对于可移动介质（如SD卡），用户可以简单地移除该存储设备。
- 它不是私有的。用户（以及其他应用）都可以访问这些文件。
- 如果用户卸载应用，您保存在通过getExternalFilesDir()获取的目录中的文件将被删除。

要使用外部存储，我们首先需要获得相应的权限。您需要使用：

- `android.permission.WRITE_EXTERNAL_STORAGE` 用于读写权限
- `android.permission.READ_EXTERNAL_STORAGE` 仅用于读取权限

要授予这些权限，您需要在AndroidManifest.xml中声明它们，如下所示

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

注意：由于它们属于危险权限，如果您使用的是API 级别 23或更高版本，您需要在运行时请求权限。

在尝试从外部存储写入或读取之前，您应始终检查存储介质是否可用。

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    // 可读写
}
if (state.equals(Environment.MEDIA_MOUNTED) ||
    state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    // 至少可读
}
```

在向外部存储写入文件时，您应决定该文件是应被识别为公共文件还是私有文件。

this code. If you **do not connect your phone**; but instead run this on **Android emulator**, your output may vary. My phone model is Coolpad Note 3 - running on Android 5.1

Storage locations on my phone:

Micro SD storage location: /storage/sdcard1

Internal Storage(UT) location: /storage/sdcard0.

Note that /sdcard & /storage/emulated/0 also point to Internal Storage(UT). But these are symlinks to /storage/sdcard0.

To clearly understand different storage paths in Android, Please go through [this answer](#)

Disclaimer: All the storage paths mentioned above are paths on **my phone**. Your files may **not** be stored on same storage paths. Because, the storage locations/paths may vary on other mobile phones depending on your vendor, manufacturer and different versions of Android OS.

Section 50.2: Using External Storage

"External" Storage is another type of storage that we can use to save files to the user's device. It has some key differences from "Internal" Storage, namely:

- It is not always available. In the case of a removable medium (SD card), the user can simply remove the storage.
- It is not private. The user (and other applications) have access to these files.
- If the user uninstalls the app, the files you save in the directory retrieved with getExternalFilesDir() will be removed.

To use External Storage, we need to first obtain the proper permissions. You will need to use:

- `android.permission.WRITE_EXTERNAL_STORAGE` for reading and writing
- `android.permission.READ_EXTERNAL_STORAGE` for just reading

To grant these permissions, you will need to identify them in your `AndroidManifest.xml` as such

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

NOTE: Since they are [Dangerous permissions](#) if you are using **API Level 23** or above, you will need to request the [permissions at runtime](#).

Before attempting to write or read from External Storage, you should always check that the storage medium is available.

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    // Available to read and write
}
if (state.equals(Environment.MEDIA_MOUNTED) ||
    state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    // Available to at least read
}
```

When writing files to the External Storage, you should decide if the file should be recognized as Public or Private.

虽然这两种类型的文件仍然可以被用户和设备上的其他应用访问，但它们之间有一个关键的区别。

公共文件在用户卸载应用时应保留在设备上。一个应保存为公共文件的例子是通过您的应用拍摄的照片。

私有文件在用户卸载应用时应全部删除。这类文件是应用特有的，对用户或其他应用没有用处。例如，您的应用下载/使用的临时文件。

以下是如何访问公共和私有文件的Documents目录的方法。

公开

```
// 访问设备公共文档目录中的应用程序目录  
File docs = new File(Environment.getExternalStoragePublicDirectory(  
    Environment.DIRECTORY_DOCUMENTS), "YourAppDirectory");  
  
// 如果目录不存在则创建该目录  
myDocs.mkdirs();
```

私有

```
// 访问应用程序的私有文档目录  
File file = new File(context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS),  
    "YourAppDirectory");  
  
// 如果目录不存在则创建该目录  
myDocs.mkdirs();
```

第50.3节：使用内部存储

默认情况下，您保存到内部存储的任何文件都是应用程序私有的。它们不能被其他应用访问，正常情况下用户也无法访问。当用户卸载应用时，这些文件会被删除。

向文件写入文本

```
String fileName= "helloworld";  
String textToWrite = "Hello, World!";  
FileOutputStream fileOutputStream;  
  
try {  
    fileOutputStream = openFileOutput(fileName, Context.MODE_PRIVATE);  
    fileOutputStream.write(textToWrite.getBytes());  
    fileOutputStream.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

向现有文件追加文本

使用Context.MODE_APPEND作为openFileOutput的模式参数

```
fileOutputStream = openFileOutput(fileName, Context.MODE_APPEND);
```

第50.4节：获取设备目录：

首先添加存储权限以读取/获取设备目录。

While both of these types of files are still accessible to the user and other applications on the device, there is a key distinction between them.

Public files should remain on the device when the user uninstalls the app. An example of a file that should be saved as Public would be photos that are taken through your application.

Private files should all be removed when the user uninstalls the app. These types of files would be app specific, and not be of use to the user or other applications. Ex. temporary files downloaded/used by your application.

Here's how to get access to the Documents directory for both Public and Private files.

Public

```
// Access your app's directory in the device's Public documents directory  
File docs = new File(Environment.getExternalStoragePublicDirectory(  
    Environment.DIRECTORY_DOCUMENTS), "YourAppDirectory");  
  
// Make the directory if it does not yet exist  
myDocs.mkdirs();
```

Private

```
// Access your app's Private documents directory  
File file = new File(context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS),  
    "YourAppDirectory");  
  
// Make the directory if it does not yet exist  
myDocs.mkdirs();
```

Section 50.3: Using Internal Storage

By default, any files that you save to Internal Storage are private to your application. They cannot be accessed by other applications, nor the user under normal circumstances. **These files are deleted when the user uninstalls the application.**

To Write Text to a File

```
String fileName= "helloworld";  
String textToWrite = "Hello, World!";  
FileOutputStream fileOutputStream;  
  
try {  
    fileOutputStream = openFileOutput(fileName, Context.MODE_PRIVATE);  
    fileOutputStream.write(textToWrite.getBytes());  
    fileOutputStream.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

To Append Text to an Existing File

Use `Context.MODE_APPEND` for the mode parameter of `openFileOutput`

```
fileOutputStream = openFileOutput(fileName, Context.MODE_APPEND);
```

Section 50.4: Fetch Device Directory :

First Add Storage permission to read/fetch device directory.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

创建模型类

```
//创建一个目录模型类
//用于在列表中存储目录标题和类型

public class DirectoryModel {
    String dirName;
    int dirType; // 设置为1或0, 0表示目录, 1表示文件。

    public int getDirType() {
        return dirType;
    }

    public void setDirType(int dirType) {
        this.dirType = dirType;
    }

    public String getDirName() {
        return dirName;
    }

    public void setDirName(String dirName) {
        this.dirName = dirName;
    }
}
```

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Create model class

```
//create one directory model class
//to store directory title and type in list

public class DirectoryModel {
    String dirName;
    int dirType; // set 1 or 0, where 0 for directory and 1 for file.

    public int getDirType() {
        return dirType;
    }

    public void setDirType(int dirType) {
        this.dirType = dirType;
    }

    public String getDirName() {
        return dirName;
    }

    public void setDirName(String dirName) {
        this.dirName = dirName;
    }
}
```

使用目录模型创建列表以添加目录数据。

```
//定义列表以显示目录

List<DirectoryModel> rootDir = new ArrayList<>();
```

使用以下方法获取目录。

```
//获取设备目录

private void getDirectory(String currDir) { // 传入设备根目录
    File f = new File(currDir);
    File[] files = f.listFiles();
    if (files != null) {
        if (files.length > 0) {
            rootDir.clear();
            for (File inFile : files) {
                if (inFile.isDirectory()) { // 如果是目录则返回true
                    // 是目录
                    DirectoryModel dir = new DirectoryModel();
                    dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                    dir.setDirType(0); // 目录类型设为0
                    rootDir.add(dir);
                } else if (inFile.isFile()) { // 如果是文件则返回true
                    // 是文件
                    DirectoryModel dir = new DirectoryModel();
                    dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                    dir.setDirType(1); // 文件类型设为1
                    rootDir.add(dir);
                }
            }
        }
    }
}
```

Create list using directory model to add directory data.

```
//define list to show directory

List<DirectoryModel> rootDir = new ArrayList<>();
```

Fetch directory using following method.

```
//to fetch device directory

private void getDirectory(String currDir) { // pass device root directory
    File f = new File(currDir);
    File[] files = f.listFiles();
    if (files != null) {
        if (files.length > 0) {
            rootDir.clear();
            for (File inFile : files) {
                if (inFile.isDirectory()) { //return true if it's directory
                    // is directory
                    DirectoryModel dir = new DirectoryModel();
                    dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                    dir.setDirType(0); // set 0 for directory
                    rootDir.add(dir);
                } else if (inFile.isFile()) { // return true if it's file
                    //is file
                    DirectoryModel dir = new DirectoryModel();
                    dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                    dir.setDirType(1); // set 1 for file
                    rootDir.add(dir);
                }
            }
        }
    }
}
```

```
    }
printDirectoryList();
}
```

在日志中打印目录列表。

```
//在日志中打印目录列表

private void printDirectoryList() {
    for (int i = 0; i < rootDir.size(); i++) {
        Log.e(TAG, "printDirectoryLogs: " + rootDir.get(i).toString());
    }
}
```

用法

```
//要获取目录，调用带根目录的函数。
```

```
String rootPath = Environment.getExternalStorageDirectory().toString(); // 返回 ==>
/storage/emulated/0/
getDirectory(rootPath );
```

要获取特定目录的内部文件/文件夹，使用相同的方法，只需更改参数，传入当前选定路径作为参数并处理相应的响应。

获取文件扩展名：

```
private String getExtension(String filename) {

    String filenameArray[] = filename.split("\\.");
    String extension = filenameArray[filenameArray.length - 1];
    Log.d(TAG, "getExtension: " + extension);

    return extension;
}
```

第50.5节：将数据库保存到SD卡（备份数据库到SD卡）

```
public static Boolean ExportDB(String DATABASE_NAME , String packageName , String folderName){
    //DATABASE_NAME 未尾包含".db"，例如"mayApp.db"
    String DBName = DATABASE_NAME.substring(0, DATABASE_NAME.length() - 3);
    File data = Environment.getDataDirectory();
    FileChannel source=null;
    FileChannel destination=null;
    String currentDBPath = "/data/" + packageName + "/databases/" + DATABASE_NAME; // 获取应用数据库
    路径

    File sd = Environment.getExternalStorageDirectory(); // 获取手机SD卡路径
    String backupPath = sd.getAbsolutePath() + folderName; // 如果你想设置备份到特定的
    文件夹名称
    /* 注意，文件夹名称必须以此开头："myFolder"。不要忘记文件夹名称开头的 "/"
```

```
你可以这样定义文件夹名："myOuterFolder/MyInnerFolder" 等等.....
*/
File dir = new File(backupPath);
if(!dir.exists()) // 如果该路径下没有文件夹，则创建它。
{
    dir.mkdirs();
```

```
    }
printDirectoryList();
}
}
```

Print directory list in log.

```
//print directory list in logs

private void printDirectoryList() {
    for (int i = 0; i < rootDir.size(); i++) {
        Log.e(TAG, "printDirectoryLogs: " + rootDir.get(i).toString());
    }
}
```

Usage

```
//to Fetch Directory Call function with root directory.
```

```
String rootPath = Environment.getExternalStorageDirectory().toString(); // return ==>
/storage/emulated/0/
getDirectory(rootPath );
```

To fetch inner files/folder of specific directory use same method just change argument, pass the current selected path in argument and handle response for same.

To get File Extension :

```
private String getExtension(String filename) {

    String filenameArray[] = filename.split("\\.");
    String extension = filenameArray[filenameArray.length - 1];
    Log.d(TAG, "getExtension: " + extension);

    return extension;
}
```

Section 50.5: Save Database on SD Card (Backup DB on SD)

```
public static Boolean ExportDB(String DATABASE_NAME , String packageName , String folderName){
    //DATABASE_NAME including ".db" at the end like "mayApp.db"
    String DBName = DATABASE_NAME.substring(0, DATABASE_NAME.length() - 3);
    File data = Environment.getDataDirectory();
    FileChannel source=null;
    FileChannel destination=null;
    String currentDBPath = "/data/" + packageName + "/databases/" + DATABASE_NAME; // getting app db
    path

    File sd = Environment.getExternalStorageDirectory(); // getting phone SD card path
    String backupPath = sd.getAbsolutePath() + folderName; // if you want to set backup in specific
    folder name
    /* be careful , foldername must initial like this : "/myFolder" . don't forget "/" at begin
    of folder name
        you could define foldername like this : "/myOuterFolder/MyInnerFolder" and so on ...
    */
    File dir = new File(backupPath);
    if(!dir.exists()) // if there was no folder at this path , it create it .
    {
        dir.mkdirs();
```

```

}

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
Date date = new Date();
/* 使用包含文件名的日期来排列它们，防止创建同名文件 */
File currentDB = new File(data, currentDBPath);
File backupDB = new File(backupPath, DBName +"("+ dateFormat.format(date)+").db");
try {
    if (currentDB.exists() && !backupDB.exists()) {
        source = new FileInputStream(currentDB).getChannel();
        destination = new FileOutputStream(backupDB).getChannel();
        destination.transferFrom(source, 0, source.size());
    }
    source.close();
    destination.close();
    return true;
}
return false;
} catch(IOException e) {
e.printStackTrace();
return false;
}
}

```

以如下方式调用此方法：

```
ExportDB("myDB.db","com.example.exam","/myFolder");
```

```

}

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
Date date = new Date();
/* use date including file name for arrange them and preventing to make file with the same*/
File currentDB = new File(data, currentDBPath);
File backupDB = new File(backupPath, DBName +"("+ dateFormat.format(date)+").db");
try {
    if (currentDB.exists() && !backupDB.exists()) {
        source = new FileInputStream(currentDB).getChannel();
        destination = new FileOutputStream(backupDB).getChannel();
        destination.transferFrom(source, 0, source.size());
        source.close();
        destination.close();
        return true;
    }
    return false;
} catch(IOException e) {
    e.printStackTrace();
    return false;
}
}

```

call this method this way :

```
ExportDB("myDB.db","com.example.exam","/myFolder");
```

第51章：安卓中的Zip文件

第51.1节：安卓上的Zip文件

```
import android.util.Log;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class Compress {
    private static final int BUFFER = 2048;

    private String[] _files;
    private String _zipFile;

    public Compress(String[] files, String zipFile) {
        _files = files;
        _zipFile = zipFile;
    }

    public void zip() {
        try {
            BufferedInputStream origin = null;
            FileOutputStream dest = new FileOutputStream(_zipFile);

            ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

            byte data[] = new byte[BUFFER];

            for(int i=0; i < _files.length; i++) {
                Log.v("Compress", "Adding: " + _files[i]);
                FileInputStream fi = new FileInputStream(_files[i]);
                origin = new BufferedInputStream(fi, BUFFER);
                ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") + 1));
                out.putNextEntry(entry);
                int count;
                while ((count = origin.read(data, 0, BUFFER)) != -1) {
                    out.write(data, 0, count);
                }
                origin.close();
            }

            out.close();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Chapter 51: Zip file in android

Section 51.1: Zip file on android

```
import android.util.Log;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class Compress {
    private static final int BUFFER = 2048;

    private String[] _files;
    private String _zipFile;

    public Compress(String[] files, String zipFile) {
        _files = files;
        _zipFile = zipFile;
    }

    public void zip() {
        try {
            BufferedInputStream origin = null;
            FileOutputStream dest = new FileOutputStream(_zipFile);

            ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

            byte data[] = new byte[BUFFER];

            for(int i=0; i < _files.length; i++) {
                Log.v("Compress", "Adding: " + _files[i]);
                FileInputStream fi = new FileInputStream(_files[i]);
                origin = new BufferedInputStream(fi, BUFFER);
                ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") + 1));
                out.putNextEntry(entry);
                int count;
                while ((count = origin.read(data, 0, BUFFER)) != -1) {
                    out.write(data, 0, count);
                }
                origin.close();
            }

            out.close();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

第52章：在安卓中解压文件

第52.1节：解压文件

```
private boolean unpackZip(String path, String zipname){  
    InputStream is;  
    ZipInputStream zis;  
    try {  
        String filename;  
        is = new FileInputStream(path + zipname);  
        zis = new ZipInputStream(new BufferedInputStream(is));  
        ZipEntry ze;  
        byte[] buffer = new byte[1024];  
        int count;  
  
        while ((ze = zis.getNextEntry()) != null){  
            // 写入文件  
            filename = ze.getName();  
  
            // 如果目录不存在，需要先创建目录，  
            // 否则会抛出异常...  
            if (ze.isDirectory()) {  
                File fmd = new File(path + filename);  
                fmd.mkdirs();  
                continue;  
            }  
  
            FileOutputStream fout = new FileOutputStream(path + filename);  
  
            // 读取zip并写入  
            while ((count = zis.read(buffer)) != -1){  
                fout.write(buffer, 0, count);  
            }  
  
            fout.close();  
            zis.closeEntry();  
        }  
  
        zis.close();  
    } catch(IOException e){  
        e.printStackTrace();  
        return false;  
    }  
  
    return true;}  
}
```

Chapter 52: Unzip File in Android

Section 52.1: Unzip file

```
private boolean unpackZip(String path, String zipname){  
    InputStream is;  
    ZipInputStream zis;  
    try {  
        String filename;  
        is = new FileInputStream(path + zipname);  
        zis = new ZipInputStream(new BufferedInputStream(is));  
        ZipEntry ze;  
        byte[] buffer = new byte[1024];  
        int count;  
  
        while ((ze = zis.getNextEntry()) != null){  
            // zapis do souboru  
            filename = ze.getName();  
  
            // Need to create directories if not exists, or  
            // it will generate an Exception...  
            if (ze.isDirectory()) {  
                File fmd = new File(path + filename);  
                fmd.mkdirs();  
                continue;  
            }  
  
            FileOutputStream fout = new FileOutputStream(path + filename);  
  
            // cteni zipu a zapis  
            while ((count = zis.read(buffer)) != -1){  
                fout.write(buffer, 0, count);  
            }  
  
            fout.close();  
            zis.closeEntry();  
        }  
  
        zis.close();  
    } catch(IOException e){  
        e.printStackTrace();  
        return false;  
    }  
  
    return true;}  
}
```

第53章：相机与图库

第53.1节：拍照

在AndroidManifest文件中添加访问相机的权限：

```
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Xml文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SurfaceView android:id="@+id/surfaceView" android:layout_height="0dip"
    android:layout_width="0dip"></SurfaceView>
<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:id="@+id/imageView"></ImageView>
</LinearLayout>
```

活动

```
import java.io.IOException;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;

public class TakePicture extends Activity implements SurfaceHolder.Callback
{
    //用于存储对 main.xml 文件中 Image View 的引用的变量
    private ImageView iv_image;
    //用于存储main.xml文件中Surface View引用的变量
    private SurfaceView sv;

    //用于显示捕获图像的位图
    private Bitmap bmp;

    //摄像头变量
    //一个Surface Holder
    private SurfaceHolder sHolder;
    //控制摄像头的变量
    private Camera mCamera;
    //摄像头参数
    private Parameters parameters;

    /** 活动首次创建时调用。 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

Chapter 53: Camera and Gallery

Section 53.1: Take photo

Add a permission to access the camera to the AndroidManifest file:

```
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Xml file :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SurfaceView android:id="@+id/surfaceView" android:layout_height="0dip"
    android:layout_width="0dip"></SurfaceView>
<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:id="@+id/imageView"></ImageView>
</LinearLayout>
```

Activity

```
import java.io.IOException;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;

public class TakePicture extends Activity implements SurfaceHolder.Callback
{
    //a variable to store a reference to the Image View at the main.xml file
    private ImageView iv_image;
    //a variable to store a reference to the Surface View at the main.xml file
    private SurfaceView sv;

    //a bitmap to display the captured image
    private Bitmap bmp;

    //Camera variables
    //a surface holder
    private SurfaceHolder sHolder;
    //a variable to control the camera
    private Camera mCamera;
    //the camera parameters
    private Parameters parameters;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.main);

//获取main.xml文件中的Image View
iv_image = (ImageView) findViewById(R.id.imageView);

//获取 main.xml 文件中的 Surface View
sv = (SurfaceView) findViewById(R.id.surfaceView);

//获取一个 surface
sHolder = sv.getHolder();

//将下面定义的回调接口方法添加为 Surface View 的回调
sHolder.addCallback(this);

//告诉 Android 该 surface 的数据将被不断替换
sHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}

@Override
public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)
{
    //获取相机参数
参数 = mCamera.getParameters();

    //设置相机参数
mCamera.setParameters(参数);
mCamera.startPreview();

    //设置拍照后应执行的代码
Camera.PictureCallback mCall = new Camera.PictureCallback()
{
    @Override
    public void onPictureTaken(byte[] 数据, Camera 相机)
    {
        //将相机获取的数据解码为Bitmap
        bmp = BitmapFactory.decodeByteArray(数据, 0, 数据.length);
        String 文件名=Environment.getExternalStorageDirectory()
            + File.separator + "testimage.jpg";
        FileOutputStream 输出流 = null;
        try {
            out = new FileOutputStream(filename);
            bmp.compress(Bitmap.CompressFormat.PNG, 100, out); // bmp 是你的 Bitmap 实例
                // PNG 是无损格式, 压缩因子 (100) 被忽略
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (out != null) {
                    out.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        // 设置 iv_image
        iv_image.setImageBitmap(bmp);
    }
};

mCamera拍照(空, 空, mCall);
}

```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.main);

//get the Image View at the main.xml file
iv_image = (ImageView) findViewById(R.id.imageView);

//get the Surface View at the main.xml file
sv = (SurfaceView) findViewById(R.id.surfaceView);

//Get a surface
sHolder = sv.getHolder();

//add the callback interface methods defined below as the Surface View callbacks
sHolder.addCallback(this);

//tells Android that this surface will have its data constantly replaced
sHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}

@Override
public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)
{
    //get camera parameters
parameters = mCamera.getParameters();

    //set camera parameters
mCamera.setParameters(parameters);
mCamera.startPreview();

    //sets what code should be executed after the picture is taken
Camera.PictureCallback mCall = new Camera.PictureCallback()
{
    @Override
    public void onPictureTaken(byte[] data, Camera camera)
    {
        //decode the data obtained by the camera into a Bitmap
        bmp = BitmapFactory.decodeByteArray(data, 0, data.length);
        String filename=Environment.getExternalStorageDirectory()
            + File.separator + "testimage.jpg";
        FileOutputStream out = null;
        try {
            out = new FileOutputStream(filename);
            bmp.compress(Bitmap.CompressFormat.PNG, 100, out); // bmp is your Bitmap
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (out != null) {
                    out.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        //set the iv_image
        iv_image.setImageBitmap(bmp);
    }
};

mCamera.takePicture(null, null, mCall);
}

```

```

}

@Override
public void surfaceCreated(SurfaceHolder 持有者)
{
    // Surface 已创建, 获取相机并告诉它预览画面绘制的位置。

mCamera = Camera.open();
try {
mCamera.setPreviewDisplay(持有者);

} catch (IOException 异常) {
    mCamera.release();
    mCamera = 空;
}
}

@Override
public void surfaceDestroyed(SurfaceHolder 持有者)
{
    //停止预览
mCamera.stopPreview();
    //释放相机
mCamera.release();
    //解绑该对象上的相机
mCamera = null;
}
}

```

```

}

@Override
public void surfaceCreated(SurfaceHolder holder)
{
    // The Surface has been created, acquire the camera and tell it where
    // to draw the preview.
mCamera = Camera.open();
try {
    mCamera.setPreviewDisplay(holder);

} catch (IOException exception) {
    mCamera.release();
    mCamera = null;
}
}

@Override
public void surfaceDestroyed(SurfaceHolder holder)
{
    //stop the preview
mCamera.stopPreview();
//release the camera
mCamera.release();
//unbind the camera from this object
mCamera = null;
}
}

```

第53.2节：从相机拍摄全尺寸照片

要拍照，首先需要在AndroidManifest.xml中声明所需权限。我们需要两个权限：

- Camera - 用于打开相机应用。如果属性required设置为true，且设备没有硬件相机，则无法安装此应用。
- WRITE_EXTERNAL_STORAGE - 该权限用于创建新文件，捕获的照片将保存在该文件中。

AndroidManifest.xml

```

<uses-feature android:name="android.hardware.camera"
              android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

```

拍摄全尺寸照片的主要思路是，在打开相机应用并拍照之前，我们需要为照片创建一个新文件。

```

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // 确保有相机应用可以处理该意图
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // 创建照片应保存的文件
        File photoFile = null;
        try {
photoFile = createImageFile();
        } catch (IOException ex) {
Log.e("DEBUG_TAG", "createFile", ex);
        }
        // 仅当文件成功创建后继续
        if (photoFile != null) {

```

AndroidManifest.xml

```

<uses-feature android:name="android.hardware.camera"
              android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

```

The main idea in taking full-sized photo from camera is that we need to create new file for photo, before we open camera app and capture photo.

```

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            Log.e("DEBUG_TAG", "createFile", ex);
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {

```

```

takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photoFile));
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}
}

private File createImageFile() throws IOException {
    // 创建一个图片文件名
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getAlbumDir();
    File image = File.createTempFile(
        imageFileName, /* 前缀 */
        ".jpg", /* 后缀 */
        storageDir /* 目录 */
    );

    // 保存文件：用于 ACTION_VIEW 意图的路径
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}

private File getAlbumDir() {
    File storageDir = null;

    if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {

        storageDir = new File(Environment.getExternalStorageDirectory()
            + "/dcim/"
            + "MyRecipes");

        if (!storageDir.mkdirs()) {
            if (!storageDir.exists()) {
                Log.d("CameraSample", "failed to create directory");
                return null;
            }
        }
    } else {
        Log.v(getString(R.string.app_name), "External storage is not mounted READ/WRITE.");
    }

    return storageDir;
}

private void setPic() {
    /* 内存不足，无法打开超过几张相机照片 */
    /* 因此预先缩放目标位图以解码文件 */

    /* 获取 ImageView 的大小 */
    int targetW = recipeImage.getWidth();
    int targetH = recipeImage.getHeight();

    /* 获取图像的大小 */
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;
}

```

```

takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photoFile));
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}
}

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getAlbumDir();
    File image = File.createTempFile(
        imageFileName, /* prefix */
        ".jpg", /* suffix */
        storageDir /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}

private File getAlbumDir() {
    File storageDir = null;

    if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {

        storageDir = new File(Environment.getExternalStorageDirectory()
            + "/dcim/"
            + "MyRecipes");

        if (!storageDir.mkdirs()) {
            if (!storageDir.exists()) {
                Log.d("CameraSample", "failed to create directory");
                return null;
            }
        }
    } else {
        Log.v(getString(R.string.app_name), "External storage is not mounted READ/WRITE.");
    }

    return storageDir;
}

private void setPic() {
    /* There isn't enough memory to open up more than a couple camera photos */
    /* So pre-scale the target bitmap into which the file is decoded */

    /* Get the size of the ImageView */
    int targetW = recipeImage.getWidth();
    int targetH = recipeImage.getHeight();

    /* Get the size of the image */
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;
}

```

```

/* 计算需要缩小的方向 */
int scaleFactor = 2;
if ((targetW > 0) && (targetH > 0)) {
    scaleFactor = Math.max(photoW / targetW, photoH / targetH);
}

/* 设置位图选项以缩放图像解码目标 */
bmOptions.inJustDecodeBounds = false;
bmOptions.inSampleSize = scaleFactor;
bmOptions.inPurgeable = true;

Matrix matrix = new Matrix();
matrix.postRotate(getRotation());

/* 将JPEG文件解码为位图 */
Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix,
false);

/* 将位图关联到ImageView */
recipeImage.setImageBitmap(bitmap);
}

private float getRotation() {
    try {
        ExifInterface ei = new ExifInterface(mCurrentPhotoPath);
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);

        switch (orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                return 90f;
            case ExifInterface.ORIENTATION_ROTATE_180:
                return 180f;
            case ExifInterface.ORIENTATION_ROTATE_270:
                return 270f;
            default:
                return 0f;
        }
    } catch (Exception e) {
        Log.e("添加配方", "获取旋转", e);
        return 0f;
    }
}

private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    sendBroadcast(mediaScanIntent);
}

private void handleBigCameraPhoto() {

    if (mCurrentPhotoPath != null) {
        setPic();
        galleryAddPic();
    }
}

```

```

/* Figure out which way needs to be reduced less */
int scaleFactor = 2;
if ((targetW > 0) && (targetH > 0)) {
    scaleFactor = Math.max(photoW / targetW, photoH / targetH);
}

/* Set bitmap options to scale the image decode target */
bmOptions.inJustDecodeBounds = false;
bmOptions.inSampleSize = scaleFactor;
bmOptions.inPurgeable = true;

Matrix matrix = new Matrix();
matrix.postRotate(getRotation());

/* Decode the JPEG file into a Bitmap */
Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix,
false);

/* Associate the Bitmap to the ImageView */
recipeImage.setImageBitmap(bitmap);
}

private float getRotation() {
    try {
        ExifInterface ei = new ExifInterface(mCurrentPhotoPath);
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);

        switch (orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                return 90f;
            case ExifInterface.ORIENTATION_ROTATE_180:
                return 180f;
            case ExifInterface.ORIENTATION_ROTATE_270:
                return 270f;
            default:
                return 0f;
        }
    } catch (Exception e) {
        Log.e("Add Recipe", "getRotation", e);
        return 0f;
    }
}

private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    sendBroadcast(mediaScanIntent);
}

private void handleBigCameraPhoto() {

    if (mCurrentPhotoPath != null) {
        setPic();
        galleryAddPic();
    }
}

```

```

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        handleBigCameraPhoto();
    }
}

```

第53.3节：正确解码从意图获取的URI中旋转的位图

```

private static final String TAG = "IntentBitmapFetch";
private static final String COLON_SEPARATOR = ":";
private static final String IMAGE = "image";

@Nullable
public Bitmap getBitmap(@NonNull Uri bitmapUri, int maxDimen) {
    InputStream is = context.getContentResolver().openInputStream(bitmapUri);
    Bitmap bitmap = BitmapFactory.decodeStream(is, null, getBitmapOptions(bitmapUri, maxDimen));

    int imgRotation = getImageRotationDegrees(bitmapUri);

    int endRotation = (imgRotation < 0) ? -imgRotation : imgRotation;
    endRotation %= 360;
    endRotation = 90 * (endRotation / 90);
    if (endRotation > 0 && bitmap != null) {
        Matrix m = new Matrix();
        m.setRotate(endRotation);
        Bitmap tmp = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), m,
                                         true);
        if (tmp != null) {
            bitmap.recycle();
            bitmap = tmp;
        }
    }

    return bitmap;
}

private BitmapFactory.Options getBitmapOptions(Uri uri, int imageMaxDimen){
    BitmapFactory.Options options = new BitmapFactory.Options();
    if (imageMaxDimen > 0) {
options.inJustDecodeBounds = true;
        decodeImage(null, uri, options);
        options.inSampleSize = calculateScaleFactor(options, imageMaxDimen);
        options.inJustDecodeBounds = false;
options.inPreferredConfig = Bitmap.Config.RGB_565;
        addInBitmapOptions(options);
    }
}

private int calculateScaleFactor(@NonNull BitmapFactory.Options bitmapOptionsMeasureOnly, int
imageMaxDimen) {
    int inSampleSize = 1;
    if (bitmapOptionsMeasureOnly.outHeight > imageMaxDimen || bitmapOptionsMeasureOnly.outWidth >
imageMaxDimen) {
        final int halfHeight = bitmapOptionsMeasureOnly.outHeight / 2;
        final int halfWidth = bitmapOptionsMeasureOnly.outWidth / 2;
        while ((halfHeight / inSampleSize) > imageMaxDimen && (halfWidth / inSampleSize) >
imageMaxDimen) {
            inSampleSize *= 2;
    }
}

```

```

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        handleBigCameraPhoto();
    }
}

```

Section 53.3: Decode bitmap correctly rotated from the uri fetched with the intent

```

private static final String TAG = "IntentBitmapFetch";
private static final String COLON_SEPARATOR = ":";
private static final String IMAGE = "image";

@Nullable
public Bitmap getBitmap(@NonNull Uri bitmapUri, int maxDimen) {
    InputStream is = context.getContentResolver().openInputStream(bitmapUri);
    Bitmap bitmap = BitmapFactory.decodeStream(is, null, getBitmapOptions(bitmapUri, maxDimen));

    int imgRotation = getImageRotationDegrees(bitmapUri);

    int endRotation = (imgRotation < 0) ? -imgRotation : imgRotation;
    endRotation %= 360;
    endRotation = 90 * (endRotation / 90);
    if (endRotation > 0 && bitmap != null) {
        Matrix m = new Matrix();
        m.setRotate(endRotation);
        Bitmap tmp = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), m,
                                         true);
        if (tmp != null) {
            bitmap.recycle();
            bitmap = tmp;
        }
    }

    return bitmap;
}

private BitmapFactory.Options getBitmapOptions(Uri uri, int imageMaxDimen){
    BitmapFactory.Options options = new BitmapFactory.Options();
    if (imageMaxDimen > 0) {
options.inJustDecodeBounds = true;
        decodeImage(null, uri, options);
        options.inSampleSize = calculateScaleFactor(options, imageMaxDimen);
        options.inJustDecodeBounds = false;
options.inPreferredConfig = Bitmap.Config.RGB_565;
        addInBitmapOptions(options);
    }
}

private int calculateScaleFactor(@NonNull BitmapFactory.Options bitmapOptionsMeasureOnly, int
imageMaxDimen) {
    int inSampleSize = 1;
    if (bitmapOptionsMeasureOnly.outHeight > imageMaxDimen || bitmapOptionsMeasureOnly.outWidth >
imageMaxDimen) {
        final int halfHeight = bitmapOptionsMeasureOnly.outHeight / 2;
        final int halfWidth = bitmapOptionsMeasureOnly.outWidth / 2;
        while ((halfHeight / inSampleSize) > imageMaxDimen && (halfWidth / inSampleSize) >
imageMaxDimen) {
            inSampleSize *= 2;
    }
}

```

```

        }
    }

    return inSampleSize;
}

public int getImageRotationDegrees(@NonNull Uri imgUri) {
    int photoRotation = ExifInterface.ORIENTATION_UNDEFINED;

    try {
        boolean hasRotation = false;
        //如果图片来自图库且不在DCIM文件夹中 (Scheme: content://)
        String[] projection = {MediaStore.Images.ImageColumns.ORIENTATION};
        Cursor cursor = context.getContentResolver().query(imgUri, projection, null, null, null);
        if (cursor != null) {
            if (cursor.getColumnCount() > 0 && cursor.moveToFirst()) {
                photoRotation = cursor.getInt(cursor.getColumnIndex(projection[0]));
                hasRotation = photoRotation != 0;
            }
            Log.d("光标方向: "+ photoRotation);
        }
        cursor.close();
    }

    //如果图像来自相机 (方案 :file://) 或来自文件夹DCIM (方案 :
    content://)
    if (!hasRotation) {
        ExifInterface exif = new ExifInterface(getAbsolutePath(imgUri));
        int exifRotation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,
                                              ExifInterface.ORIENTATION_NORMAL);
        switch (exifRotation) {
            case ExifInterface.ORIENTATION_ROTATE_90: {
                photoRotation = 90;
                break;
            }
            case ExifInterface.ORIENTATION_ROTATE_180: {
                photoRotation = 180;
                break;
            }
            case ExifInterface.ORIENTATION_ROTATE_270: {
                photoRotation = 270;
                break;
            }
        }
        Log.d(TAG, "Exif orientation: "+ photoRotation);
    }
} catch (IOException e) {
    Log.e(TAG, "Error determining rotation for image"+ imgUri, e);
}
return photoRotation;
}

@TargetApi(Build.VERSION_CODES.KITKAT)
private String getAbsolutePath(Uri uri) {
    //代码片段编辑自: http://stackoverflow.com/a/20559418/2235133
    String filePath = uri.getPath();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(context, uri)) {
        // 将返回 "image:x"
        String[] wholeID = TextUtils.split(DocumentsContract.getDocumentId(uri), COLON_SEPARATOR);
        // 在冒号处分割, 使用数组中的第二项
        String type = wholeID[0];
        if (!IMAGE.equalsIgnoreCase(type)) {//如果不是图片类型, 意味着它来自远程位置, 比如谷歌相册

```

```

        }
    }

    return inSampleSize;
}

public int getImageRotationDegrees(@NonNull Uri imgUri) {
    int photoRotation = ExifInterface.ORIENTATION_UNDEFINED;

    try {
        boolean hasRotation = false;
        //If image comes from the gallery and is not in the folder DCIM (Scheme: content://)
        String[] projection = {MediaStore.Images.ImageColumns.ORIENTATION};
        Cursor cursor = context.getContentResolver().query(imgUri, projection, null, null, null);
        if (cursor != null) {
            if (cursor.getColumnCount() > 0 && cursor.moveToFirst()) {
                photoRotation = cursor.getInt(cursor.getColumnIndex(projection[0]));
                hasRotation = photoRotation != 0;
                Log.d("Cursor orientation: "+ photoRotation);
            }
            cursor.close();
        }

        //If image comes from the camera (Scheme: file://) or is from the folder DCIM (Scheme:
        content://)
        if (!hasRotation) {
            ExifInterface exif = new ExifInterface(getAbsolutePath(imgUri));
            int exifRotation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,
                                                  ExifInterface.ORIENTATION_NORMAL);
            switch (exifRotation) {
                case ExifInterface.ORIENTATION_ROTATE_90: {
                    photoRotation = 90;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_180: {
                    photoRotation = 180;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_270: {
                    photoRotation = 270;
                    break;
                }
            }
            Log.d(TAG, "Exif orientation: "+ photoRotation);
        }
    } catch (IOException e) {
        Log.e(TAG, "Error determining rotation for image"+ imgUri, e);
    }
    return photoRotation;
}

@TargetApi(Build.VERSION_CODES.KITKAT)
private String getAbsolutePath(Uri uri) {
    //Code snippet edited from: http://stackoverflow.com/a/20559418/2235133
    String filePath = uri.getPath();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(context, uri)) {
        // Will return "image:x"
        String[] wholeID = TextUtils.split(DocumentsContract.getDocumentId(uri), COLON_SEPARATOR);
        // Split at colon, use second item in the array
        String type = wholeID[0];
        if (!IMAGE.equalsIgnoreCase(type)) {//If it not type image, it means it comes from a remote
location, like Google Photos

```

```

String id = wholeID[1];
String[] column = {MediaStore.Images.Media.DATA};
// 其中 id 等于
String sel = MediaStore.Images.Media._ID + "=?";
Cursor cursor = context.getContentResolver();
query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
      column, sel, new String[]{id}, null);
if (cursor != null) {
    int columnIndex = cursor.getColumnIndex(column[0]);
    if (cursor.moveToFirst()) {
        filePath = cursor.getString(columnIndex);
    }
    cursor.close();
}
Log.d(TAG, "获取到的URI的绝对路径" + uri);
}
return filePath;
}

```

第53.4节：设置相机分辨率

以编程方式设置高分辨率。

```

Camera mCamera = Camera.open();
Camera.Parameters params = mCamera.getParameters();

// 检查你的相机支持哪些分辨率
List<Size> sizes = params.getSupportedPictureSizes();

// 遍历所有可用分辨率并选择一个。
// 选定的分辨率将存储在 mSize 中。
尺寸 mSize;
for (Size size : sizes) {
    Log.i(TAG, "可用分辨率: "+size.width+" "+size.height);
    mSize = size;
}

Log.i(TAG, "选择的分辨率: "+mSize.width+" "+mSize.height);
params.setPictureSize(mSize.width, mSize.height);
mCamera.setParameters(params);

```

第53.5节：如何启动相机或图库并将相机结果保存到存储

首先你需要 Uri 和临时文件夹以及请求码：

```

public final int REQUEST_SELECT_PICTURE = 0x01;
public final int REQUEST_CODE_TAKE_PICTURE = 0x2;
public static String TEMP_PHOTO_FILE_NAME = "photo_";
Uri mImageCaptureUri;
File mFileTemp;

```

然后初始化 mFileTemp：

```

public void initTempFile(){
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {

```

```

String id = wholeID[1];
String[] column = {MediaStore.Images.Media.DATA};
// where id is equal to
String sel = MediaStore.Images.Media._ID + "=?";
Cursor cursor = context.getContentResolver();
query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
      column, sel, new String[]{id}, null);
if (cursor != null) {
    int columnIndex = cursor.getColumnIndex(column[0]);
    if (cursor.moveToFirst()) {
        filePath = cursor.getString(columnIndex);
    }
    cursor.close();
}
Log.d(TAG, "Fetched absolute path for uri" + uri);
}
return filePath;
}

```

Section 53.4: Set camera resolution

Set High resolution programmatically.

```

Camera mCamera = Camera.open();
Camera.Parameters params = mCamera.getParameters();

// Check what resolutions are supported by your camera
List<Size> sizes = params.getSupportedPictureSizes();

// Iterate through all available resolutions and choose one.
// The chosen resolution will be stored in mSize.
Size mSize;
for (Size size : sizes) {
    Log.i(TAG, "Available resolution: "+size.width+" "+size.height);
    mSize = size;
}

Log.i(TAG, "Chosen resolution: "+mSize.width+" "+mSize.height);
params.setPictureSize(mSize.width, mSize.height);
mCamera.setParameters(params);

```

Section 53.5: How to start camera or gallery and save camera result to storage

First of all you need Uri and temp Folders and request codes :

```

public final int REQUEST_SELECT_PICTURE = 0x01;
public final int REQUEST_CODE_TAKE_PICTURE = 0x2;
public static String TEMP_PHOTO_FILE_NAME = "photo_";
Uri mImageCaptureUri;
File mFileTemp;

```

Then init mFileTemp :

```

public void initTempFile(){
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {

```

```

mFileTemp = new File(Environment.getExternalStorageDirectory() + File.separator
    +getResources().getString(R.string.app_foldername) + File.separator
    +getResources().getString(R.string.pictures_folder)
    , TEMP_PHOTO_FILE_NAME
    + System.currentTimeMillis() + ".jpg");
mFileTemp.getParentFile().mkdirs();
} else {
mFileTemp = new File(getFilesDir() + File.separator
    +getResources().getString(R.string.app_foldername)
    + File.separator + getResources().getString(R.string.pictures_folder)
    , TEMP_PHOTO_FILE_NAME + System.currentTimeMillis() + ".jpg");
mFileTemp.getParentFile().mkdirs();
}
}

```

打开相机和图库意图：

```

public void openCamera(){
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
try {
mImageCaptureUri = null;
String state = Environment.getExternalStorageState();
if (Environment.MEDIA_MOUNTED.equals(state)) {
    mImageCaptureUri = Uri.fromFile(mFileTemp);
} else {
mImageCaptureUri = InternalStorageContentProvider.CONTENT_URI;

}
intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageCaptureUri);
intent.putExtra("return-data", true);
startActivityForResult(intent, REQUEST_CODE_TAKE_PICTURE);
} catch (Exception e) {

Log.d("error", "cannot take picture", e);
}
}

public void openGallery(){
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN
    && ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE)
    != PackageManager.PERMISSION_GRANTED) {
requestPermission(Manifest.permission.READ_EXTERNAL_STORAGE,
    getString(R.string.permission_read_storage_rationale),
    REQUEST_STORAGE_READ_ACCESS_PERMISSION);
} else {
Intent intent = new Intent();
intent.setType("image/*");
intent.setAction(Intent.ACTION_GET_CONTENT);
intent.addCategory(Intent.CATEGORY_OPENABLE);
startActivityForResult(Intent.createChooser(intent, getString(R.string.select_image)),
REQUEST_SELECT_PICTURE);
}
}

```

然后在 onActivityResult 方法中：

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

```

```

mFileTemp = new File(Environment.getExternalStorageDirectory() + File.separator
    +getResources().getString(R.string.app_foldername) + File.separator
    +getResources().getString(R.string.pictures_folder)
    , TEMP_PHOTO_FILE_NAME
    + System.currentTimeMillis() + ".jpg");
mFileTemp.getParentFile().mkdirs();
} else {
mFileTemp = new File(getFilesDir() + File.separator
    +getResources().getString(R.string.app_foldername)
    + File.separator + getResources().getString(R.string.pictures_folder)
    , TEMP_PHOTO_FILE_NAME + System.currentTimeMillis() + ".jpg");
mFileTemp.getParentFile().mkdirs();
}
}

```

Opening Camera and Gallery intents :

```

public void openCamera(){
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
try {
mImageCaptureUri = null;
String state = Environment.getExternalStorageState();
if (Environment.MEDIA_MOUNTED.equals(state)) {
    mImageCaptureUri = Uri.fromFile(mFileTemp);
} else {
mImageCaptureUri = InternalStorageContentProvider.CONTENT_URI;

}
intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageCaptureUri);
intent.putExtra("return-data", true);
startActivityForResult(intent, REQUEST_CODE_TAKE_PICTURE);
} catch (Exception e) {

Log.d("error", "cannot take picture", e);
}
}

public void openGallery(){
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN
    && ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE)
    != PackageManager.PERMISSION_GRANTED) {
requestPermission(Manifest.permission.READ_EXTERNAL_STORAGE,
    getString(R.string.permission_read_storage_rationale),
    REQUEST_STORAGE_READ_ACCESS_PERMISSION);
} else {
Intent intent = new Intent();
intent.setType("image/*");
intent.setAction(Intent.ACTION_GET_CONTENT);
intent.addCategory(Intent.CATEGORY_OPENABLE);
startActivityForResult(Intent.createChooser(intent, getString(R.string.select_image)),
REQUEST_SELECT_PICTURE);
}
}

```

Then in onActivityResult method :

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

```

```

if (resultCode != RESULT_OK) {
    return;
}
Bitmap bitmap;

switch (requestCode) {

    case REQUEST_SELECT_PICTURE:
        try {
Uri uri = data.getData();
        try {
bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
            Bitmap bitmapScaled = Bitmap.createScaledBitmap(bitmap, 800, 800, true);
            Drawable drawable=new BitmapDrawable(bitmapScaled);
mImage.setImageDrawable(drawable);
            mImage.setVisibility(View.VISIBLE);
        } catch (IOException e) {
Log.v("act result", "there is an error : "+e.getContent());
        }
    } catch (Exception e) {
Log.v("act result", "there is an error : "+e.getContent());
    }
    break;
    case REQUEST_CODE_TAKE_PICTURE:
        try{
Bitmap bitmappicture = MediaStore.Images.Media.getBitmap(getContentResolver() ,
mImageCaptureUri);
mImage.setImageBitmap(bitmappicture);
            mImage.setVisibility(View.VISIBLE);
        }catch (IOException e){
Log.v("错误相机",e.getMessage());
        }
    break;
}
super.onActivityResult(requestCode, resultCode, data);
}

```

您需要在AndroidManifest.xml中声明以下权限：

```

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />

```

并且您需要处理运行时权限，例如读取/写入外部存储等...

我在openGallery方法中检查READ_EXTERNAL_STORAGE权限：

我的requestPermission方法：

```

protected void requestPermission(final String permission, String rationale, final int requestCode)
{
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
        showDialog(getString(R.string.permission_title_rationale), rationale,
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    ActivityCompat.requestPermissions(BasePermissionActivity.this,
                        new String[]{permission}, requestCode);
                }
            }, getString(android.R.string.ok), null, getString(android.R.string.cancel));
    } else {

```

```

if (resultCode != RESULT_OK) {
    return;
}
Bitmap bitmap;

switch (requestCode) {

    case REQUEST_SELECT_PICTURE:
        try {
Uri uri = data.getData();
        try {
bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
            Bitmap bitmapScaled = Bitmap.createScaledBitmap(bitmap, 800, 800, true);
            Drawable drawable=new BitmapDrawable(bitmapScaled);
mImage.setImageDrawable(drawable);
            mImage.setVisibility(View.VISIBLE);
        } catch (IOException e) {
Log.v("act result", "there is an error : "+e.getContent());
        }
    } catch (Exception e) {
Log.v("act result", "there is an error : "+e.getContent());
    }
    break;
    case REQUEST_CODE_TAKE_PICTURE:
        try{
Bitmap bitmappicture = MediaStore.Images.Media.getBitmap(getContentResolver() ,
mImageCaptureUri);
mImage.setImageBitmap(bitmappicture);
            mImage.setVisibility(View.VISIBLE);
        }catch (IOException e){
Log.v("error camera",e.getMessage());
        }
    break;
}
super.onActivityResult(requestCode, resultCode, data);
}

```

You need theese permissions in AndroidManifest.xml:

```

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />

```

And you need to handle [runtime permissions](#) such as Read/Write external storage etc ...

I am checking READ_EXTERNAL_STORAGE permission in my openGallery method:

My requestPermission method :

```

protected void requestPermission(final String permission, String rationale, final int requestCode)
{
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
        showDialog(getString(R.string.permission_title_rationale), rationale,
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    ActivityCompat.requestPermissions(BasePermissionActivity.this,
                        new String[]{permission}, requestCode);
                }
            }, getString(android.R.string.ok), null, getString(android.R.string.cancel));
    } else {

```

```
ActivityCompat.requestPermissions(this, new String[]{permission}, requestCode);
}
}
```

然后重写 onRequestPermissionsResult 方法：

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    switch (requestCode) {
        case REQUEST_STORAGE_READ_ACCESS_PERMISSION:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                handleGallery();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

showAlertDialog 方法：

```
protected void showAlertDialog(@Nullable String title, @Nullable String message,
                               @Nullable DialogInterface.OnClickListener
onPositiveButtonClickListener,
                               @NonNull String positiveText,
                               @Nullable DialogInterface.OnClickListener
onNegativeButtonClickListener,
                               @NonNull String negativeText) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton(positiveText, onPositiveButtonClickListener);
    builder.setNegativeButton(negativeText, onNegativeButtonClickListener);
    mAlertDialog = builder.show();
}
```

```
ActivityCompat.requestPermissions(this, new String[]{permission}, requestCode);
}
}
```

Then Override onRequestPermissionsResult method :

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    switch (requestCode) {
        case REQUEST_STORAGE_READ_ACCESS_PERMISSION:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                handleGallery();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

showAlertDialog method :

```
protected void showAlertDialog(@Nullable String title, @Nullable String message,
                               @Nullable DialogInterface.OnClickListener
onPositiveButtonClickListener,
                               @NonNull String positiveText,
                               @Nullable DialogInterface.OnClickListener
onNegativeButtonClickListener,
                               @NonNull String negativeText) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton(positiveText, onPositiveButtonClickListener);
    builder.setNegativeButton(negativeText, onNegativeButtonClickListener);
    mAlertDialog = builder.show();
}
```

第54章：Camera 2 API

参数	详细信息
CameraCaptureSession	为 CameraDevice 配置的捕获会话，用于从摄像头捕获图像或重新处理之前在同一会话中捕获的图像 摄像头或重新处理之前在同一会话中捕获的图像
摄像头设备	连接到安卓设备的单个摄像头的表示
CameraCharacteristics	描述 CameraDevice 的属性。这些属性对于给定的 CameraDevice 是固定的，可以通过 CameraManager 接口使用 getCameraCharacteristics(String)
CameraManager	用于检测、描述和连接 CameraDevices 的系统服务管理器。 可以通过调用 Context.getSystemService() 获取该类的实例
CaptureRequest	捕获单张图像所需的设置和输出的不可变包，来自 摄像头设备。包含捕获硬件（传感器、镜头、闪光灯）、 处理管线、控制算法和输出缓冲区的配置。还包含 用于此次捕获的目标 Surface 列表。可以通过使用 CaptureRequest.Builder 实例创建，该实例通过调用 createCaptureRequest(int)
CaptureResult	来自图像传感器的单次图像捕获结果的子集。包含捕获硬件（传感器、镜头、闪光灯）、 处理管线、控制算法和输出缓冲区的最终配置子集。它由 CameraDevice 在处理 CaptureRequest 后生成

第54.1节：在TextureView中预览主摄像头

在这种情况下，针对 API 23 构建，因此权限也会被处理。

您必须在清单文件中添加以下权限（无论您使用的 API 级别是多少）：

```
<uses-permission android:name="android.permission.CAMERA"/>
```

我们即将创建一个活动 (Camera2Activity.java)，它将用设备摄像头的预览填充一个 TextureView。

我们将使用的活动是一个典型的 AppCompatActivity：

```
public class Camera2Activity extends AppCompatActivity {
```

属性（您可能需要阅读完整示例才能理解部分内容）

Camera2 API 保证的 MAX_PREVIEW_SIZE 为 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;  
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

TextureView.SurfaceTextureListener 处理 TextureView 上的多个生命周期事件。在本例中，我们监听这些事件。
当 SurfaceTexture 准备好时，我们初始化相机。当其尺寸发生变化时，我们相应地设置来自相机的预览。

```
private final TextureView.SurfaceTextureListener mSurfaceTextureListener  
= new TextureView.SurfaceTextureListener() {  
  
    @Override  
    public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int height) {  
        openCamera(width, height);  
    }  
}
```

Chapter 54: Camera 2 API

Parameter	Details
CameraCaptureSession	A configured capture session for a CameraDevice, used for capturing images from the camera or reprocessing images captured from the camera in the same session previously
CameraDevice	A representation of a single camera connected to an Android device
CameraCharacteristics	The properties describing a CameraDevice. These properties are fixed for a given CameraDevice, and can be queried through the CameraManager interface with getCameraCharacteristics(String)
CameraManager	A system service manager for detecting, characterizing, and connecting to CameraDevices. You can get an instance of this class by calling Context.getSystemService()
CaptureRequest	An immutable package of settings and outputs needed to capture a single image from the camera device. Contains the configuration for the capture hardware (sensor, lens, flash), the processing pipeline, the control algorithms, and the output buffers. Also contains the list of target Surfaces to send image data to for this capture. Can be created by using a CaptureRequest.Builder instance, obtained by calling createCaptureRequest(int)
CaptureResult	The subset of the results of a single image capture from the image sensor. Contains a subset of the final configuration for the capture hardware (sensor, lens, flash), the processing pipeline, the control algorithms, and the output buffers. It is produced by a CameraDevice after processing a CaptureRequest

Section 54.1: Preview the main camera in a TextureView

In this case, building against API 23, so permissions are handled too.

You must add in the Manifest the following permission (wherever the API level you're using):

```
<uses-permission android:name="android.permission.CAMERA"/>
```

We're about to create an activity (Camera2Activity.java) that fills a TextureView with the preview of the device's camera.

The Activity we're going to use is a typical AppCompatActivity:

```
public class Camera2Activity extends AppCompatActivity {
```

Attributes (You may need to read the entire example to understand some of it)

The MAX_PREVIEW_SIZE guaranteed by Camera2 API is 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;  
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

TextureView.SurfaceTextureListener handles several lifecycle events on a TextureView. In this case, we're listening to those events. When the SurfaceTexture is ready, we initialize the camera. When its size changes, we setup the preview coming from the camera accordingly

```
private final TextureView.SurfaceTextureListener mSurfaceTextureListener  
= new TextureView.SurfaceTextureListener() {  
  
    @Override  
    public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int height) {  
        openCamera(width, height);  
    }  
}
```

```

@Override
public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width, int height) {
    configureTransform(width, height);
}

@Override
public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
    return true;
}

@Override
public void onSurfaceTextureUpdated(SurfaceTexture texture) {
}

};

```

CameraDevice 表示一个物理设备的相机。在此属性中，我们保存当前CameraDevice的ID。

```
private String mCameraId;
```

这是我们用来“绘制”相机预览的视图（TextureView）

```
private TextureView mTextureView;
```

用于相机预览的CameraCaptureSession

```
private CameraCaptureSession mCaptureSession;
```

已打开的CameraDevice的引用

```
private CameraDevice mCameraDevice;
```

相机预览的Size（尺寸）

```
private Size mPreviewSize;
```

当CameraDevice状态改变时调用的CameraDevice.StateCallback

```

private final CameraDevice.StateCallback mStateCallback = new CameraDevice.StateCallback() {

    @Override
    public void onOpened(@NonNull CameraDevice cameraDevice) {
        // 当相机打开时调用此方法。我们在这里开始相机预览。
        mCameraOpenCloseLock.release();
        mCameraDevice = cameraDevice;
        createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice cameraDevice) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
    }

    @Override
    public void onError(@NonNull CameraDevice cameraDevice, int error) {

```

```

@Override
public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width, int height) {
    configureTransform(width, height);
}

@Override
public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
    return true;
}

@Override
public void onSurfaceTextureUpdated(SurfaceTexture texture) {
}

};

```

A CameraDevice represent one physical device's camera. In this attribute, we save the ID of the current CameraDevice

```
private String mCameraId;
```

This is the view (TextureView) that we'll be using to "draw" the preview of the Camera

```
private TextureView mTextureView;
```

The CameraCaptureSession for camera preview

```
private CameraCaptureSession mCaptureSession;
```

A reference to the opened CameraDevice

```
private CameraDevice mCameraDevice;
```

The Size of camera preview.

```
private Size mPreviewSize;
```

CameraDevice.StateCallback is called when CameraDevice changes its state

```

private final CameraDevice.StateCallback mStateCallback = new CameraDevice.StateCallback() {

    @Override
    public void onOpened(@NonNull CameraDevice cameraDevice) {
        // This method is called when the camera is opened. We start camera preview here.
        mCameraOpenCloseLock.release();
        mCameraDevice = cameraDevice;
        createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice cameraDevice) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
    }

    @Override
    public void onError(@NonNull CameraDevice cameraDevice, int error) {

```

```
mCameraOpenCloseLock.release();
    cameraDevice.close();
mCameraDevice = null;
    finish();
}

};
```

用于运行不应阻塞用户界面的任务的额外线程

```
private HandlerThread mBackgroundThread;
```

用于在后台运行任务的Handler

```
private Handler mBackgroundHandler;
```

处理静态图像捕获的ImageReader

```
private ImageReader mImageReader;
```

用于相机预览的CaptureRequest.Builder

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

由mPreviewRequestBuilder生成的CaptureRequest

```
private CaptureRequest mPreviewRequest;
```

防止应用在关闭相机前退出的Semaphore

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

权限请求的常量ID

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Android 生命周期方法

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera2);

    mTextureView = (TextureView) findViewById(R.id.texture);
}

@Override
public void onResume() {
    super.onResume();
    startBackgroundThread();

    // 当屏幕关闭后再打开时, SurfaceTexture 已经可用, "onSurfaceTextureAvailable" 不会被调用。在这种情况下, // 我们可以从这里打开摄像头并开始预览 (否则, 我们会等待 SurfaceTextureListener 中的 surface 准备好)。

    if (mTextureView.isAvailable()) {
        openCamera(mTextureView.getWidth(), mTextureView.getHeight());
    }
}
```

```
mCameraOpenCloseLock.release();
    cameraDevice.close();
mCameraDevice = null;
    finish();
}

};
```

An additional thread for running tasks that shouldn't block the UI

```
private HandlerThread mBackgroundThread;
```

A Handler for running tasks in the background

```
private Handler mBackgroundHandler;
```

An ImageReader that handles still image capture

```
private ImageReader mImageReader;
```

CaptureRequest.Builder for the camera preview

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

CaptureRequest generated by mPreviewRequestBuilder

```
private CaptureRequest mPreviewRequest;
```

A Semaphore to prevent the app from exiting before closing the camera.

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

Constant ID of the permission request

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Android Lifecycle methods

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera2);

    mTextureView = (TextureView) findViewById(R.id.texture);
}

@Override
public void onResume() {
    super.onResume();
    startBackgroundThread();

    // When the screen is turned off and turned back on, the SurfaceTexture is already
    // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can open
    // a camera and start preview from here (otherwise, we wait until the surface is ready in
    // the SurfaceTextureListener).
    if (mTextureView.isAvailable()) {
        openCamera(mTextureView.getWidth(), mTextureView.getHeight());
    }
}
```

```

} else {
mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
}
}

@Override
public void onPause() {
closeCamera();
stopBackgroundThread();
super.onPause();
}

```

Camera2 相关方法

这些都是使用 Camera2 API 的方法

```

private void openCamera(int width, int height) {
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
!= PackageManager.PERMISSION_GRANTED) {
requestCameraPermission();
return;
}
setUpCameraOutputs(width, height);
configureTransform(width, height);
CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
try {
if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
throw new RuntimeException("等待锁定相机打开超时。");
}
manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
} catch (CameraAccessException e) {
e.printStackTrace();
} catch (InterruptedException e) {
throw new RuntimeException("尝试锁定相机打开时被中断。", e);
}
}

```

关闭当前相机

```

private void closeCamera() {
try {
mCameraOpenCloseLock.acquire();
if (null != mCaptureSession) {
mCaptureSession.close();
mCaptureSession = null;
}
if (null != mCameraDevice) {
mCameraDevice.close();
mCameraDevice = null;
}
if (null != mImageReader) {
mImageReader.close();
mImageReader = null;
}
} catch (InterruptedException e) {
throw new RuntimeException("尝试锁定相机关闭时被中断。", e);
} finally {
mCameraOpenCloseLock.release();
}
}

```

```

} else {
mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
}
}

@Override
public void onPause() {
closeCamera();
stopBackgroundThread();
super.onPause();
}

```

Camera2 related methods

These are methods that uses the Camera2 APIs

```

private void openCamera(int width, int height) {
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
!= PackageManager.PERMISSION_GRANTED) {
requestCameraPermission();
return;
}
setUpCameraOutputs(width, height);
configureTransform(width, height);
CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
try {
if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
throw new RuntimeException("Time out waiting to lock camera opening.");
}
manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
} catch (CameraAccessException e) {
e.printStackTrace();
} catch (InterruptedException e) {
throw new RuntimeException("Interrupted while trying to lock camera opening.", e);
}
}

```

Closes the current camera

```

private void closeCamera() {
try {
mCameraOpenCloseLock.acquire();
if (null != mCaptureSession) {
mCaptureSession.close();
mCaptureSession = null;
}
if (null != mCameraDevice) {
mCameraDevice.close();
mCameraDevice = null;
}
if (null != mImageReader) {
mImageReader.close();
mImageReader = null;
}
} catch (InterruptedException e) {
throw new RuntimeException("Interrupted while trying to lock camera closing.", e);
} finally {
mCameraOpenCloseLock.release();
}
}

```

设置与相机相关的成员变量

```

私有无返回值 setUpCameraOutputs(int 宽度, int 高度) {
    CameraManager 管理器 = (CameraManager) 获得SystemService(Context.CAMERA_SERVICE);
    尝试 {
        对于 (String 相机ID : 管理器.获取相机ID列表()) {
            CameraCharacteristics 特性
                = 管理器.获取相机特性(相机ID);

            // 本示例中不使用前置摄像头。
            Integer 朝向 = 特性.获取(CameraCharacteristics.LENS_FACING);
            如果 (朝向 != 空 && 朝向 == CameraCharacteristics.LENS_FACING_FRONT) {
                继续;
            }

        StreamConfigurationMap 映射 = 特性.获取(
            CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
        如果 (映射 == 空) {
            继续;
        }

        // 对于静态图像捕捉, 我们使用可用的最大尺寸。
        最大尺寸 = Collections.max(
            Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
            new CompareSizesByArea());
        mImageReader = ImageReader.newInstance(largest.getWidth(), largest.getHeight(),
            ImageFormat.JPEG, /*maxImages*/2);
        mImageReader.setOnImageAvailableListener(
            null, mBackgroundHandler);

        Point displaySize = new Point();
        getWindowManager().getDefaultDisplay().getSize(displaySize);
        int rotatedPreviewWidth = width;
        int rotatedPreviewHeight = height;
        int maxPreviewWidth = displaySize.x;
        int maxPreviewHeight = displaySize.y;

        if (maxPreviewWidth > MAX_PREVIEW_WIDTH) {
            maxPreviewWidth = MAX_PREVIEW_WIDTH;
        }

        if (maxPreviewHeight > MAX_PREVIEW_HEIGHT) {
            maxPreviewHeight = MAX_PREVIEW_HEIGHT;
        }

        // 危险! 尝试使用过大的预览尺寸可能会超过摄像头
        // 总线的带宽限制, 导致预览效果很好但捕获的数据
        // 是垃圾数据。
        mPreviewSize = chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
            rotatedPreviewWidth, rotatedPreviewHeight, maxPreviewWidth,
            maxPreviewHeight, largest);

        mCameraId = cameraId;
        return;
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (NullPointerException e) {
        // 当前当使用Camera2API但设备不支持时, 会抛出NPE异常。
        Toast.makeText(Camera2Activity.this, "此设备不支持Camera2 API",
            Toast.LENGTH_LONG).show();
    }
}

```

Sets up member variables related to camera

```

private void setUpCameraOutputs(int width, int height) {
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        for (String cameraId : manager.getCameraIdList()) {
            CameraCharacteristics characteristics
                = manager.getCameraCharacteristics(cameraId);

            // We don't use a front facing camera in this sample.
            Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
            if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
                continue;
            }

            StreamConfigurationMap map = characteristics.get(
                CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
            if (map == null) {
                continue;
            }

            // For still image captures, we use the largest available size.
            Size largest = Collections.max(
                Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
                new CompareSizesByArea());
            mImageReader = ImageReader.newInstance(largest.getWidth(), largest.getHeight(),
                ImageFormat.JPEG, /*maxImages*/2);
            mImageReader.setOnImageAvailableListener(
                null, mBackgroundHandler);

            Point displaySize = new Point();
            getWindowManager().getDefaultDisplay().getSize(displaySize);
            int rotatedPreviewWidth = width;
            int rotatedPreviewHeight = height;
            int maxPreviewWidth = displaySize.x;
            int maxPreviewHeight = displaySize.y;

            if (maxPreviewWidth > MAX_PREVIEW_WIDTH) {
                maxPreviewWidth = MAX_PREVIEW_WIDTH;
            }

            if (maxPreviewHeight > MAX_PREVIEW_HEIGHT) {
                maxPreviewHeight = MAX_PREVIEW_HEIGHT;
            }

            // Danger! Attempting to use too large a preview size could exceed the camera
            // bus' bandwidth limitation, resulting in gorgeous previews but the storage of
            // garbage capture data.
            mPreviewSize = chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
                rotatedPreviewWidth, rotatedPreviewHeight, maxPreviewWidth,
                maxPreviewHeight, largest);

            mCameraId = cameraId;
            return;
        }
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (NullPointerException e) {
        // Currently an NPE is thrown when the Camera2API is used but not supported on the
        // device this code runs.
        Toast.makeText(Camera2Activity.this, "Camera2 API not supported on this device",
            Toast.LENGTH_LONG).show();
    }
}

```

```
}
```

```
}
```

创建一个新的CameraCaptureSession用于相机预览

```
private void createCameraPreviewSession() {
    try {
SurfaceTexture 纹理 = mTextureView.getSurfaceTexture();
    assert 纹理 != null;

        // 我们将默认缓冲区的大小配置为所需的相机预览大小。
纹理.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight());

        // 这是我们启动预览所需的输出 Surface。
Surface 表面 = new Surface(纹理);

        // 我们使用输出 Surface 设置 CaptureRequest.Builder。
mPreviewRequestBuilder
    = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
mPreviewRequestBuilder.addTarget(表面);

        // 这里，我们为相机预览创建一个 CameraCaptureSession。
mCameraDevice.createCaptureSession(Arrays.asList(surface, mImageReader.getSurface()),
    new CameraCaptureSession.StateCallback() {

        @Override
        public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
            // 相机已关闭
            if (null == mCameraDevice) {
                return;
            }

            // 会话准备好后，我们开始显示预览。
mCaptureSession = cameraCaptureSession;
            try {
                // 相机预览时自动对焦应为连续模式。
mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,
    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);

                // 最后，我们开始显示摄像头预览。
mPreviewRequest = mPreviewRequestBuilder.build();
                mCaptureSession.setRepeatingRequest(mPreviewRequest,
                    null, mBackgroundHandler);
            } catch (CameraAccessException e) {
e.printStackTrace();
            }
        }

        @Override
        public void onConfigureFailed(
@NonNull CameraCaptureSession cameraCaptureSession) {
            showToast("Failed");
        }
    }, null
);
    } catch (CameraAccessException e) {
e.printStackTrace();
    }
}
```

```
}
```

```
}
```

Creates a new CameraCaptureSession for camera preview

```
private void createCameraPreviewSession() {
    try {
SurfaceTexture texture = mTextureView.getSurfaceTexture();
    assert texture != null;

        // We configure the size of default buffer to be the size of camera preview we want.
texture.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight());

        // This is the output Surface we need to start preview.
Surface surface = new Surface(texture);

        // We set up a CaptureRequest.Builder with the output Surface.
mPreviewRequestBuilder
    = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
mPreviewRequestBuilder.addTarget(surface);

        // Here, we create a CameraCaptureSession for camera preview.
mCameraDevice.createCaptureSession(Arrays.asList(surface, mImageReader.getSurface()),
    new CameraCaptureSession.StateCallback() {

        @Override
        public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
            // The camera is already closed
            if (null == mCameraDevice) {
                return;
            }

            // When the session is ready, we start displaying the preview.
mCaptureSession = cameraCaptureSession;
            try {
                // Auto focus should be continuous for camera preview.
mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,
    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);

                // Finally, we start displaying the camera preview.
mPreviewRequest = mPreviewRequestBuilder.build();
                mCaptureSession.setRepeatingRequest(mPreviewRequest,
                    null, mBackgroundHandler);
            } catch (CameraAccessException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onConfigureFailed(
@NonNull CameraCaptureSession cameraCaptureSession) {
            showToast("Failed");
        }
    }, null
);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

权限相关方法 适用于 Android API 23 及以上版本

Permissions related methods For Android API 23+

```

private void requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA)) {
        new AlertDialog.Builder(Camera2Activity.this)
            .setMessage("R.string.request_permission")
            .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    ActivityCompat.requestPermissions(Camera2Activity.this,
                        new String[]{Manifest.permission.CAMERA},
                        REQUEST_CAMERA_PERMISSION);
                }
            })
            .setNegativeButton(android.R.string.cancel,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                    }
                })
            .create();
    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
            REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] != PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(Camera2Activity.this, "错误：未授予相机权限",
                Toast.LENGTH_LONG).show();
        } else {
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}

```

后台线程 / 处理器方法

```

private void startBackgroundThread() {
    mBackgroundThread = new HandlerThread("CameraBackground");
    mBackgroundThread.start();
    mBackgroundHandler = new Handler(mBackgroundThread.getLooper());
}

private void stopBackgroundThread() {
    mBackgroundThread.quitSafely();
    try {
        mBackgroundThread.join();
        mBackgroundThread = null;
        mBackgroundHandler = null;
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

工具方法

```

private void requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA)) {
        new AlertDialog.Builder(Camera2Activity.this)
            .setMessage("R.string.request_permission")
            .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    ActivityCompat.requestPermissions(Camera2Activity.this,
                        new String[]{Manifest.permission.CAMERA},
                        REQUEST_CAMERA_PERMISSION);
                }
            })
            .setNegativeButton(android.R.string.cancel,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                    }
                })
            .create();
    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
            REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] != PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(Camera2Activity.this, "ERROR: Camera permissions not granted",
                Toast.LENGTH_LONG).show();
        } else {
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}

```

Background thread / handler methods

```

private void startBackgroundThread() {
    mBackgroundThread = new HandlerThread("CameraBackground");
    mBackgroundThread.start();
    mBackgroundHandler = new Handler(mBackgroundThread.getLooper());
}

private void stopBackgroundThread() {
    mBackgroundThread.quitSafely();
    try {
        mBackgroundThread.join();
        mBackgroundThread = null;
        mBackgroundHandler = null;
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

Utility methods

给定相机支持的尺寸选项，选择一个最小的尺寸，该尺寸至少与相应的纹理视图尺寸一样大，且最多与相应的最大尺寸一样大，并且其宽高比与指定值匹配。如果不存在，则选择一个最大的尺寸，该尺寸最多与相应的最大尺寸一样大，且其宽高比与指定值匹配

```
private static Size 选择最优尺寸(Size[] 选项, int 纹理视图宽度,
                                  int 纹理视图高度, int 最大宽度, int 最大高度, Size
                                  宽高比) {

    // 收集支持的分辨率，这些分辨率至少与预览Surface一样大
    List<Size> 足够大 = new ArrayList<>();
    // 收集支持的分辨率，这些分辨率小于预览Surface
    List<Size> 不够大 = new ArrayList<>();
    int w = 宽高比.getWidth();
    int h = 宽高比.getHeight();
    for (Size 选项 : 选择) {
        if (选项.getWidth() <= 最大宽度 && 选项.getHeight() <= 最大高度 &&
            选项.getHeight() == 选项.getWidth() * h / w) {
            if (option.getWidth() >= textureViewWidth &&
                option.getHeight() >= textureViewHeight) {
                bigEnough.add(option);
            } else {
                notBigEnough.add(option);
            }
        }
    }

    // 选择那些足够大的中最小的。如果没有足够大的，则选择那些不够大的中最大的。
    if (bigEnough.size() > 0) {
        return Collections.min(bigEnough, new CompareSizesByArea());
    } else if (notBigEnough.size() > 0) {
        return Collections.max(notBigEnough, new CompareSizesByArea());
    } else {
        Log.e("Camera2", "找不到合适的预览尺寸");
        return choices[0];
    }
}
```

此方法配置必要的Matrix变换到mTextureView

```
private void configureTransform(int viewWidth, int viewHeight) {
    if (null == mTextureView || null == mPreviewSize) {
        return;
    }
    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    Matrix matrix = new Matrix();
    RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
    RectF bufferRect = new RectF(0, 0, mPreviewSize.getHeight(), mPreviewSize.getWidth());
    float centerX = viewRect.centerX();
    float centerY = viewRect.centerY();
    if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
        bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
        matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
        float scale = Math.max(
            (float) viewHeight / mPreviewSize.getHeight(),
            (float) viewWidth / mPreviewSize.getWidth());
        matrix.postScale(scale, scale, centerX, centerY);
        matrix.postRotate(90 * (rotation - 2), centerX, centerY);
    } else if (Surface.ROTATION_180 == rotation) {
```

Given choices of Sizes supported by a camera, choose the smallest one that is at least as large as the respective texture view size, and that is as most as large as the respective max size, and whose aspect ratio matches with the specified value. If doesn't exist, choose the largest one that is at most as large as the respective max size, and whose aspect ratio matches with the specified value

```
private static Size chooseOptimalSize(Size[] choices, int textureViewWidth,
                                      int textureViewHeight, int maxWidth, int maxHeight, Size
                                      aspectRatio) {

    // Collect the supported resolutions that are at least as big as the preview Surface
    List<Size> bigEnough = new ArrayList<>();
    // Collect the supported resolutions that are smaller than the preview Surface
    List<Size> notBigEnough = new ArrayList<>();
    int w = aspectRatio.getWidth();
    int h = aspectRatio.getHeight();
    for (Size option : choices) {
        if (option.getWidth() <= maxWidth && option.getHeight() <= maxHeight &&
            option.getHeight() == option.getWidth() * h / w) {
            if (option.getWidth() >= textureViewWidth &&
                option.getHeight() >= textureViewHeight) {
                bigEnough.add(option);
            } else {
                notBigEnough.add(option);
            }
        }
    }

    // Pick the smallest of those big enough. If there is no one big enough, pick the
    // largest of those not big enough.
    if (bigEnough.size() > 0) {
        return Collections.min(bigEnough, new CompareSizesByArea());
    } else if (notBigEnough.size() > 0) {
        return Collections.max(notBigEnough, new CompareSizesByArea());
    } else {
        Log.e("Camera2", "Couldn't find any suitable preview size");
        return choices[0];
    }
}
```

This method configures the necessary Matrix transformation to mTextureView

```
private void configureTransform(int viewWidth, int viewHeight) {
    if (null == mTextureView || null == mPreviewSize) {
        return;
    }
    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    Matrix matrix = new Matrix();
    RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
    RectF bufferRect = new RectF(0, 0, mPreviewSize.getHeight(), mPreviewSize.getWidth());
    float centerX = viewRect.centerX();
    float centerY = viewRect.centerY();
    if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
        bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
        matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
        float scale = Math.max(
            (float) viewHeight / mPreviewSize.getHeight(),
            (float) viewWidth / mPreviewSize.getWidth());
        matrix.postScale(scale, scale, centerX, centerY);
        matrix.postRotate(90 * (rotation - 2), centerX, centerY);
    } else if (Surface.ROTATION_180 == rotation) {
```

```
matrix.postRotate(180, centerX, centerY);
}
mTextureView.setTransform(matrix);
}
```

此方法基于面积比较两个Size对象。

```
static class CompareSizesByArea implements Comparator<Size> {

    @Override
    public int compare(Size lhs, Size rhs) {
        // 我们在此处进行强制转换以确保乘法不会溢出
        return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
            (long) rhs.getWidth() * rhs.getHeight());
    }
}
```

这里没什么可看的

```
/***
 * 在UI线程上显示{@link Toast}。
 *
 * @param text 要显示的消息
 */
private void showToast(final String text) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(Camera2Activity.this, text, Toast.LENGTH_SHORT).show();
        }
    });
}
```

```
matrix.postRotate(180, centerX, centerY);
}
mTextureView.setTransform(matrix);
}
```

This method compares two Sizes based on their areas.

```
static class CompareSizesByArea implements Comparator<Size> {

    @Override
    public int compare(Size lhs, Size rhs) {
        // We cast here to ensure the multiplications won't overflow
        return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
            (long) rhs.getWidth() * rhs.getHeight());
    }
}
```

Not much to see here

```
/***
 * Shows a {@link Toast} on the UI thread.
 *
 * @param text The message to show
 */
private void showToast(final String text) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(Camera2Activity.this, text, Toast.LENGTH_SHORT).show();
        }
    });
}
```

第55章：Android中的指纹API

第55.1节：如何使用Android指纹API保存用户密码

此示例辅助类与指纹管理器交互，并执行密码的加密和解密。请注意，本示例中用于加密的方法是AES。这并不是唯一的加密方式，其他示例也存在。在本示例中，数据的加密和解密方式如下：

加密：

1. 用户向辅助类提供所需的未加密密码。
2. 用户需要提供指纹。
3. 认证通过后，辅助类从KeyStore获取密钥，并使用Cipher加密密码。
4. 密码和IV盐（IV每次加密时重新生成，不重复使用）被保存到共享偏好中，以便稍后解密时使用。

解密：

1. 用户请求解密密码。
2. 用户需要提供指纹。
3. 助手使用初始化向量（IV）构建一个Cipher，一旦用户通过身份验证，KeyStore便从中获取密钥 KeyStore 并解密密码。

```
public class FingerPrintAuthHelper {  
  
    private static final String FINGER_PRINT_HELPER = "FingerPrintAuthHelper";  
    private static final String ENCRYPTED_PASS_SHARED_PREF_KEY = "ENCRYPTED_PASS_SHARED_PREF_KEY";  
    private static final String LAST_USED_IV_SHARED_PREF_KEY = "LAST_USED_IV_SHARED_PREF_KEY";  
    private static final String MY_APP_ALIAS = "MY_APP_ALIAS";  
  
    private KeyguardManager keyguardManager;  
    private FingerprintManager fingerprintManager;  
  
    private final Context context;  
    private KeyStore keyStore;  
    private KeyGenerator keyGenerator;  
  
    private String lastError;  
  
    public interface Callback {  
        void onSuccess(String savedPass);  
  
        void onFailure(String message);  
  
        void onHelp(int helpCode, String helpString);  
    }  
  
    public FingerPrintAuthHelper(Context context) {  
        this.context = context;  
    }  
  
    public String getLastErrorMessage() {  
        return lastError;  
    }  
}
```

Chapter 55: Fingerprint API in android

Section 55.1: How to use Android Fingerprint API to save user passwords

This example helper class interacts with the finger print manager and performs encryption and decryption of password. Please note that the method used for encryption in this example is AES. This is not the only way to encrypt and other examples exist. In this example the data is encrypted and decrypted in the following manner:

Encryption:

1. User gives helper the desired non-encrypted password.
2. User is required to provide fingerprint.
3. Once authenticated, the helper obtains a key from the [KeyStore](#) and encrypts the password using a Cipher.
4. Password and IV salt (IV is recreated for every encryption and is not reused) are saved to shared preferences to be used later in the decryption process.

Decryption:

1. User requests to decrypt the password.
2. User is required to provide fingerprint.
3. The helper builds a Cipher using the IV and once user is authenticated, the KeyStore obtains a key from the [KeyStore](#) and deciphers the password.

```
public class FingerPrintAuthHelper {  
  
    private static final String FINGER_PRINT_HELPER = "FingerPrintAuthHelper";  
    private static final String ENCRYPTED_PASS_SHARED_PREF_KEY = "ENCRYPTED_PASS_SHARED_PREF_KEY";  
    private static final String LAST_USED_IV_SHARED_PREF_KEY = "LAST_USED_IV_SHARED_PREF_KEY";  
    private static final String MY_APP_ALIAS = "MY_APP_ALIAS";  
  
    private KeyguardManager keyguardManager;  
    private FingerprintManager fingerprintManager;  
  
    private final Context context;  
    private KeyStore keyStore;  
    private KeyGenerator keyGenerator;  
  
    private String lastError;  
  
    public interface Callback {  
        void onSuccess(String savedPass);  
  
        void onFailure(String message);  
  
        void onHelp(int helpCode, String helpString);  
    }  
  
    public FingerPrintAuthHelper(Context context) {  
        this.context = context;  
    }  
  
    public String getLastErrorMessage() {  
        return lastError;  
    }  
}
```

```

@TargetApi(Build.VERSION_CODES.M)
public boolean init() {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        setError("该安卓版本不支持指纹认证");
        return false;
    }

    keyguardManager = (KeyguardManager) context.getSystemService(KEYGUARD_SERVICE);
    fingerprintManager = (FingerprintManager) context.getSystemService(FINGERPRINT_SERVICE);

    if (!keyguardManager.isKeyguardSecure()) {
        setError("用户未启用锁屏");
        return false;
    }

    if (!hasPermission()) {
        setError("用户未授予使用指纹的权限");
        return false;
    }

    if (!fingerprintManager.hasEnrolledFingerprints()) {
        setError("用户未注册任何指纹");
        return false;
    }

    if (!initKeyStore()) {
        return false;
    }
    return false;
}

@Nullable
@RequiresApi(api = Build.VERSION_CODES.M)
private Cipher createCipher(int mode) throws NoSuchPaddingException, NoSuchAlgorithmException,
UnrecoverableKeyException, KeyStoreException, InvalidKeyException,
InvalidAlgorithmParameterException {
    Cipher cipher = Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/" +
        KeyProperties.BLOCK_MODE_CBC + "/" +
        KeyProperties.ENCRYPTION_PADDING_PKCS7);

    Key key = keyStore.getKey(MY_APP_ALIAS, null);
    if (key == null) {
        return null;
    }
    if(mode == Cipher.ENCRYPT_MODE) {
        cipher.init(mode, key);
        byte[] iv = cipher.getIV();
        saveIv(iv);
    } else {
        byte[] lastIv = getLastIv();
        cipher.init(mode, key, new IvParameterSpec(lastIv));
    }
    return cipher;
}

@NonNull
@RequiresApi(api = Build.VERSION_CODES.M)
private KeyGenParameterSpec createKeyGenParameterSpec() {
    return new KeyGenParameterSpec.Builder(MY_APP_ALIAS, KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
.setBlockModes(KeyProperties.BLOCK_MODE_CBC)
.setUserAuthenticationRequired(true)

```

```

@TargetApi(Build.VERSION_CODES.M)
public boolean init() {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        setError("This Android version does not support fingerprint authentication");
        return false;
    }

    keyguardManager = (KeyguardManager) context.getSystemService(KEYGUARD_SERVICE);
    fingerprintManager = (FingerprintManager) context.getSystemService(FINGERPRINT_SERVICE);

    if (!keyguardManager.isKeyguardSecure()) {
        setError("User hasn't enabled Lock Screen");
        return false;
    }

    if (!hasPermission()) {
        setError("User hasn't granted permission to use Fingerprint");
        return false;
    }

    if (!fingerprintManager.hasEnrolledFingerprints()) {
        setError("User hasn't registered any fingerprints");
        return false;
    }

    if (!initKeyStore()) {
        return false;
    }
    return false;
}

@Nullable
@RequiresApi(api = Build.VERSION_CODES.M)
private Cipher createCipher(int mode) throws NoSuchPaddingException, NoSuchAlgorithmException,
UnrecoverableKeyException, KeyStoreException, InvalidKeyException,
InvalidAlgorithmParameterException {
    Cipher cipher = Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/" +
        KeyProperties.BLOCK_MODE_CBC + "/" +
        KeyProperties.ENCRYPTION_PADDING_PKCS7);

    Key key = keyStore.getKey(MY_APP_ALIAS, null);
    if (key == null) {
        return null;
    }
    if(mode == Cipher.ENCRYPT_MODE) {
        cipher.init(mode, key);
        byte[] iv = cipher.getIV();
        saveIv(iv);
    } else {
        byte[] lastIv = getLastIv();
        cipher.init(mode, key, new IvParameterSpec(lastIv));
    }
    return cipher;
}

@NonNull
@RequiresApi(api = Build.VERSION_CODES.M)
private KeyGenParameterSpec createKeyGenParameterSpec() {
    return new KeyGenParameterSpec.Builder(MY_APP_ALIAS, KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
.setBlockModes(KeyProperties.BLOCK_MODE_CBC)
.setUserAuthenticationRequired(true)

```

```

.setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
    .build();
}

@RequiresApi(api = Build.VERSION_CODES.M)
private boolean initKeyStore() {
    try {
        keyStore = KeyStore.getInstance("AndroidKeyStore");
        keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
        "AndroidKeyStore");
        keyStore.load(null);
        if (getLastIv() == null) {
            KeyGenParameterSpec keyGeneratorSpec = createKeyGenParameterSpec();
            keyGenerator.init(keyGeneratorSpec);
            keyGenerator.generateKey();
        }
    } 捕获 (Throwable t) {
        setError("初始化密钥库和密钥生成器失败: " + t.getMessage());
        return false;
    }
    return true;
}

@RequiresApi(api = Build.VERSION_CODES.M)
private void authenticate(CancellationSignal cancellationSignal,
FingerPrintAuthenticationListener authListener, int mode) {
    try {
        if (hasPermission()) {
            Cipher cipher = createCipher(mode);
            FingerprintManager.CryptoObject crypto = new
            FingerprintManager.CryptoObject(cipher);
            fingerprintManager.authenticate(crypto, cancellationSignal, 0, authListener, null);
        } else {
            authListener.getCallback().onFailure("用户未授权使用指纹");
        }
    } 捕获 (Throwable t) {
        authListener.getCallback().onFailure("发生错误: " + t.getMessage());
    }
}

private String getSavedEncryptedPassword() {
    SharedPreferences sharedpreferences = getSharedPreferences();
    if (sharedpreferences != null) {
        return sharedpreferences.getString(ENCRYPTED_PASS_SHARED_PREF_KEY, null);
    }
    return null;
}

private void saveEncryptedPassword(String encryptedPassword) {
    SharedPreferences.Editor edit = getSharedPreferences().edit();
    edit.putString(ENCRYPTED_PASS_SHARED_PREF_KEY, encryptedPassword);
    edit.commit();
}

private byte[] getLastIv() {
    SharedPreferences sharedpreferences = getSharedPreferences();
    if (sharedpreferences != null) {
        String ivString = sharedpreferences.getString(LAST_USED_IV_SHARED_PREF_KEY, null);
        if (ivString != null) {
            return decodeBytes(ivString);
        }
    }
}

```

```

.setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
    .build();
}

@RequiresApi(api = Build.VERSION_CODES.M)
private boolean initKeyStore() {
    try {
        keyStore = KeyStore.getInstance("AndroidKeyStore");
        keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
        "AndroidKeyStore");
        keyStore.load(null);
        if (getLastIv() == null) {
            KeyGenParameterSpec keyGeneratorSpec = createKeyGenParameterSpec();
            keyGenerator.init(keyGeneratorSpec);
            keyGenerator.generateKey();
        }
    } catch (Throwable t) {
        setError("Failed init of keyStore & keyGenerator: " + t.getMessage());
        return false;
    }
    return true;
}

@RequiresApi(api = Build.VERSION_CODES.M)
private void authenticate(CancellationSignal cancellationSignal,
FingerPrintAuthenticationListener authListener, int mode) {
    try {
        if (hasPermission()) {
            Cipher cipher = createCipher(mode);
            FingerprintManager.CryptoObject crypto = new
            FingerprintManager.CryptoObject(cipher);
            fingerprintManager.authenticate(crypto, cancellationSignal, 0, authListener, null);
        } else {
            authListener.getCallback().onFailure("User hasn't granted permission to use
            Fingerprint");
        }
    } catch (Throwable t) {
        authListener.getCallback().onFailure("An error occurred: " + t.getMessage());
    }
}

private String getSavedEncryptedPassword() {
    SharedPreferences sharedpreferences = getSharedPreferences();
    if (sharedpreferences != null) {
        return sharedpreferences.getString(ENCRYPTED_PASS_SHARED_PREF_KEY, null);
    }
    return null;
}

private void saveEncryptedPassword(String encryptedPassword) {
    SharedPreferences.Editor edit = getSharedPreferences().edit();
    edit.putString(ENCRYPTED_PASS_SHARED_PREF_KEY, encryptedPassword);
    edit.commit();
}

private byte[] getLastIv() {
    SharedPreferences sharedpreferences = getSharedPreferences();
    if (sharedpreferences != null) {
        String ivString = sharedpreferences.getString(LAST_USED_IV_SHARED_PREF_KEY, null);
        if (ivString != null) {
            return decodeBytes(ivString);
        }
    }
}

```

```

    }
}

private void saveIv(byte[] iv) {
SharedPreferences.Editor edit = getSharedPreferences().edit();
    String string = encodeBytes(iv);
edit.putString(LAST_USED_IV_SHARED_PREF_KEY, string);
    edit.commit();
}

private SharedPreferences getSharedPreferences() {
    return context.getSharedPreferences(FINGER_PRINT_HELPER, 0);
}

@RequiresApi(api = Build.VERSION_CODES.M)
private boolean hasPermission() {
    return ActivityCompat.checkSelfPermission(context, Manifest.permission.USE_FINGERPRINT) ==
PackageManager.PERMISSION_GRANTED;
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void savePassword(@NonNull String password, CancellationSignal cancellationSignal,
Callback callback) {
authenticate(cancellationSignal, new FingerPrintEncryptPasswordListener(callback,
password), Cipher.ENCRYPT_MODE);
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void getPassword(CancellationSignal cancellationSignal, Callback callback) {
    authenticate(cancellationSignal, new FingerPrintDecryptPasswordListener(callback),
Cipher.DECRYPT_MODE);
}

@RequiresApi(api = Build.VERSION_CODES.M)
public boolean encryptPassword(Cipher cipher, String password) {
    try {
        // Encrypt the text
        if(password.isEmpty()) {
            setError("密码为空");
            return false;
        }

        if (cipher == null) {
setError("无法创建加密器");
            return false;
        }

        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream, cipher);
        byte[] bytes = password.getBytes(Charset.defaultCharset());
        cipherOutputStream.write(bytes);
        cipherOutputStream.flush();
        cipherOutputStream.close();
        saveEncryptedPassword(encodeBytes(outputStream.toByteArray()));
    } catch (Throwable t) {
setError("加密失败 " + t.getMessage());
            return false;
    }

    return true;
}

```

```

    }
}

private void saveIv(byte[] iv) {
SharedPreferences.Editor edit = getSharedPreferences().edit();
    String string = encodeBytes(iv);
edit.putString(LAST_USED_IV_SHARED_PREF_KEY, string);
    edit.commit();
}

private SharedPreferences getSharedPreferences() {
    return context.getSharedPreferences(FINGER_PRINT_HELPER, 0);
}

@RequiresApi(api = Build.VERSION_CODES.M)
private boolean hasPermission() {
    return ActivityCompat.checkSelfPermission(context, Manifest.permission.USE_FINGERPRINT) ==
PackageManager.PERMISSION_GRANTED;
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void savePassword(@NonNull String password, CancellationSignal cancellationSignal,
Callback callback) {
    authenticate(cancellationSignal, new FingerPrintEncryptPasswordListener(callback,
password), Cipher.ENCRYPT_MODE);
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void getPassword(CancellationSignal cancellationSignal, Callback callback) {
    authenticate(cancellationSignal, new FingerPrintDecryptPasswordListener(callback),
Cipher.DECRYPT_MODE);
}

@RequiresApi(api = Build.VERSION_CODES.M)
public boolean encryptPassword(Cipher cipher, String password) {
    try {
        // Encrypt the text
        if(password.isEmpty()) {
            setError("Password is empty");
            return false;
        }

        if (cipher == null) {
            setError("Could not create cipher");
            return false;
        }

        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream, cipher);
        byte[] bytes = password.getBytes(Charset.defaultCharset());
        cipherOutputStream.write(bytes);
        cipherOutputStream.flush();
        cipherOutputStream.close();
        saveEncryptedPassword(encodeBytes(outputStream.toByteArray()));
    } catch (Throwable t) {
        setError("Encryption failed " + t.getMessage());
            return false;
    }

    return true;
}

```

```

}

private byte[] decodeBytes(String s) {
    final int len = s.length();

    // "111" 不是有效的十六进制编码。
    if( len%2 != 0 )
        throw new IllegalArgumentException("hexBinary 需要偶数长度: "+s);

    byte[] out = new byte[len/2];

    for( int i=0; i<len; i+=2 ) {
        int h = hexToBin(s.charAt(i));
        int l = hexToBin(s.charAt(i+1));
        if( h== -1 || l== -1 )
            throw new IllegalArgumentException("contains illegal character for hexBinary: "+s);

        out[i/2] = (byte)(h*16+l);
    }

    return out;
}

private static int hexToBin( char ch ) {
    if( '0'<=ch && ch<='9' )    return ch-'0';
    if( 'A'<=ch && ch<='F' )    return ch-'A'+10;
    if( 'a'<=ch && ch<='f' )    return ch-'a'+10;
    return -1;
}

private static final char[] hexCode = "0123456789ABCDEF".toCharArray();

public String encodeBytes(byte[] data) {
StringBuilder r = new StringBuilder(data.length*2);
    for ( byte b : data) {
r.append(hexCode[(b >> 4) & 0xF]);
r.append(hexCode[(b & 0xF)]);
    }
    return r.toString();
}

@NonNull
private String decipher(Cipher cipher) throws IOException, IllegalBlockSizeException,
BadPaddingException {
    String retVal = null;
    String savedEncryptedPassword = getSavedEncryptedPassword();
    if (savedEncryptedPassword != null) {
        byte[] decodedPassword = decodeBytes(savedEncryptedPassword);
        CipherInputStream cipherInputStream = new CipherInputStream(new
ByteArrayInputStream(decodedPassword), cipher);

ArrayList<Byte> values = new ArrayList<>();
        int nextByte;
        while ((nextByte = cipherInputStream.read()) != -1) {
            values.add((byte) nextByte);
        }
cipherInputStream.close();

        byte[] bytes = new byte[values.size()];
        for (int i = 0; i < values.size(); i++) {
            bytes[i] = values.get(i).byteValue();
        }
    }
}

```

```

}

private byte[] decodeBytes(String s) {
    final int len = s.length();

    // "111" is not a valid hex encoding.
    if( len%2 != 0 )
        throw new IllegalArgumentException("hexBinary needs to be even-length: "+s);

    byte[] out = new byte[len/2];

    for( int i=0; i<len; i+=2 ) {
        int h = hexToBin(s.charAt(i));
        int l = hexToBin(s.charAt(i+1));
        if( h== -1 || l== -1 )
            throw new IllegalArgumentException("contains illegal character for hexBinary: "+s);

        out[i/2] = (byte)(h*16+l);
    }

    return out;
}

private static int hexToBin( char ch ) {
    if( '0'<=ch && ch<='9' )    return ch-'0';
    if( 'A'<=ch && ch<='F' )    return ch-'A'+10;
    if( 'a'<=ch && ch<='f' )    return ch-'a'+10;
    return -1;
}

private static final char[] hexCode = "0123456789ABCDEF".toCharArray();

public String encodeBytes(byte[] data) {
StringBuilder r = new StringBuilder(data.length*2);
    for ( byte b : data) {
r.append(hexCode[(b >> 4) & 0xF]);
r.append(hexCode[(b & 0xF)]);
    }
    return r.toString();
}

@NonNull
private String decipher(Cipher cipher) throws IOException, IllegalBlockSizeException,
BadPaddingException {
    String retVal = null;
    String savedEncryptedPassword = getSavedEncryptedPassword();
    if (savedEncryptedPassword != null) {
        byte[] decodedPassword = decodeBytes(savedEncryptedPassword);
        CipherInputStream cipherInputStream = new CipherInputStream(new
ByteArrayInputStream(decodedPassword), cipher);

ArrayList<Byte> values = new ArrayList<>();
        int nextByte;
        while ((nextByte = cipherInputStream.read()) != -1) {
            values.add((byte) nextByte);
        }
cipherInputStream.close();

        byte[] bytes = new byte[values.size()];
        for (int i = 0; i < values.size(); i++) {
            bytes[i] = values.get(i).byteValue();
        }
    }
}

```

```

    retVal = new String(bytes, Charset.defaultCharset());
}
return retVal;
}

private void setError(String error) {
    lastError = error;
    Log.w(FINGER_PRINT_HELPER, lastError);
}

@RequiresApi(Build.VERSION_CODES.M)
protected class FingerPrintAuthenticationListener extends
FingerprintManager.AuthenticationCallback {

    protected final Callback callback;

    public FingerPrintAuthenticationListener(@NonNull Callback callback) {
        this.callback = callback;
    }

    public void onAuthenticationError(int errorCode, CharSequence errString) {
        callback.onFailure("Authentication error [" + errorCode + "] " + errString);
    }

    /**
     * 当认证过程中遇到可恢复的错误时调用。帮助字符串用于向用户提供错误原因的指导，例如
     * “传感器脏了，请清洁。”
     * @param helpCode 一个标识错误信息的整数
     * @param helpString 一个可在界面显示的可读字符串
     */
    public void onAuthenticationHelp(int helpCode, CharSequence helpString) {
        callback.onHelp(helpCode, helpString.toString());
    }

    /**
     * 当指纹被识别时调用。
     * @param result 一个包含认证相关数据的对象
     */
    public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
    }

    /**
     * 当指纹有效但未被识别时调用。
     */
    public void onAuthenticationFailed() {
        callback.onFailure("认证失败");
    }

    public @NonNull
    Callback getCallback() {
        return callback;
    }
}

@RequiresApi(api = Build.VERSION_CODES.M)
private class FingerPrintEncryptPasswordListener extends FingerPrintAuthenticationListener {

    private final String password;

    public FingerPrintEncryptPasswordListener(Callback callback, String password) {

```

```

        retVal = new String(bytes, Charset.defaultCharset());
    }
    return retVal;
}

private void setError(String error) {
    lastError = error;
    Log.w(FINGER_PRINT_HELPER, lastError);
}

@RequiresApi(Build.VERSION_CODES.M)
protected class FingerPrintAuthenticationListener extends
FingerprintManager.AuthenticationCallback {

    protected final Callback callback;

    public FingerPrintAuthenticationListener(@NonNull Callback callback) {
        this.callback = callback;
    }

    public void onAuthenticationError(int errorCode, CharSequence errString) {
        callback.onFailure("Authentication error [" + errorCode + "] " + errString);
    }

    /**
     * Called when a recoverable error has been encountered during authentication. The help
     * string is provided to give the user guidance for what went wrong, such as
     * "Sensor dirty, please clean it."
     * @param helpCode An integer identifying the error message
     * @param helpString A human-readable string that can be shown in UI
     */
    public void onAuthenticationHelp(int helpCode, CharSequence helpString) {
        callback.onHelp(helpCode, helpString.toString());
    }

    /**
     * Called when a fingerprint is recognized.
     * @param result An object containing authentication-related data
     */
    public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
    }

    /**
     * Called when a fingerprint is valid but not recognized.
     */
    public void onAuthenticationFailed() {
        callback.onFailure("Authentication failed");
    }

    public @NonNull
    Callback getCallback() {
        return callback;
    }
}

@RequiresApi(api = Build.VERSION_CODES.M)
private class FingerPrintEncryptPasswordListener extends FingerPrintAuthenticationListener {

    private final String password;

    public FingerPrintEncryptPasswordListener(Callback callback, String password) {

```

```

super(callback);
this.password = password;
}

public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
    Cipher cipher = result.getCryptoObject().getCipher();
    try {
        if (encryptPassword(cipher, password)) {
            callback.onSuccess("Encrypted");
        } else {
            callback.onFailure("Encryption failed");
        }
    } catch (Exception e) {
        callback.onFailure("Encryption failed " + e.getMessage());
    }
}

@RequiresApi(Build.VERSION_CODES.M)
protected class FingerPrintDecryptPasswordListener extends FingerPrintAuthenticationListener {

    public FingerPrintDecryptPasswordListener(@NonNull Callback callback) {
        super(callback);
    }

    public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
        Cipher cipher = result.getCryptoObject().getCipher();
        try {
            String savedPass = decipher(cipher);
            if (savedPass != null) {
                callback.onSuccess(savedPass);
            } else {
                callback.onFailure("Failed deciphering");
            }
        } catch (Exception e) {
            callback.onFailure("Deciphering failed " + e.getMessage());
        }
    }
}

```

下面的这个活动是一个非常基础的示例，展示了如何获取用户保存的密码并与辅助工具交互。

```

public class MainActivity extends AppCompatActivity {

    private TextView passwordTextView;
    private FingerPrintAuthHelper fingerPrintAuthHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        passwordTextView = (TextView) findViewById(R.id.password);
        errorTextView = (TextView) findViewById(R.id.error);

        View setPasswordButton = findViewById(R.id.set_password_button);
        setPasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

```

```

super(callback);
this.password = password;
}

public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
    Cipher cipher = result.getCryptoObject().getCipher();
    try {
        if (encryptPassword(cipher, password)) {
            callback.onSuccess("Encrypted");
        } else {
            callback.onFailure("Encryption failed");
        }
    } catch (Exception e) {
        callback.onFailure("Encryption failed " + e.getMessage());
    }
}

@RequiresApi(Build.VERSION_CODES.M)
protected class FingerPrintDecryptPasswordListener extends FingerPrintAuthenticationListener {

    public FingerPrintDecryptPasswordListener(@NonNull Callback callback) {
        super(callback);
    }

    public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
        Cipher cipher = result.getCryptoObject().getCipher();
        try {
            String savedPass = decipher(cipher);
            if (savedPass != null) {
                callback.onSuccess(savedPass);
            } else {
                callback.onFailure("Failed deciphering");
            }
        } catch (Exception e) {
            callback.onFailure("Deciphering failed " + e.getMessage());
        }
    }
}

```

This activity below is a very basic example of how to get a user saved password and interact with the helper.

```

public class MainActivity extends AppCompatActivity {

    private TextView passwordTextView;
    private FingerPrintAuthHelper fingerPrintAuthHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        passwordTextView = (TextView) findViewById(R.id.password);
        errorTextView = (TextView) findViewById(R.id.error);

        View setPasswordButton = findViewById(R.id.set_password_button);
        setPasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

```

```

fingerPrintAuthHelper.savePassword(passwordTextView.getText().toString(), new
CancellationSignal(), getAuthListener(false));
}

});

View getPasswordButton = findViewById(R.id.get_password_button);
getPasswordButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
fingerPrintAuthHelper.getPassword(new CancellationSignal(),
getAuthListener(true));
}
}
});

// 启动指纹助手。如果失败则向用户显示错误
private void startFingerPrintAuthHelper() {
fingerPrintAuthHelper = new FingerPrintAuthHelper(this);
if (!fingerPrintAuthHelper.init()) {
errorTextView.setText(fingerPrintAuthHelper.getLastError());
}
}

@NonNull
private FingerPrintAuthHelper.Callback getAuthListener(final boolean isGetPass) {
return new FingerPrintAuthHelper.Callback() {
@Override
public void onSuccess(String result) {
if (isGetPass) {
errorTextView.setText("成功！！！密码 = " + result);
} else {
errorTextView.setText("加密密码 = " + result);
}
}

@Override
public void onFailure(String message) {
errorTextView.setText("失败 - " + message);
}

@Override
public void onHelp(int helpCode, String helpString) {
errorTextView.setText("需要帮助 - " + helpString);
}
};
}
}

```

```

fingerPrintAuthHelper.savePassword(passwordTextView.getText().toString(), new
CancellationSignal(), getAuthListener(false));
}

);

View getPasswordButton = findViewById(R.id.get_password_button);
getPasswordButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
fingerPrintAuthHelper.getPassword(new CancellationSignal(),
getAuthListener(true));
}
}
});

// Start the finger print helper. In case this fails show error to user
private void startFingerPrintAuthHelper() {
fingerPrintAuthHelper = new FingerPrintAuthHelper(this);
if (!fingerPrintAuthHelper.init()) {
errorTextView.setText(fingerPrintAuthHelper.getLastError());
}
}

@NonNull
private FingerPrintAuthHelper.Callback getAuthListener(final boolean isGetPass) {
return new FingerPrintAuthHelper.Callback() {
@Override
public void onSuccess(String result) {
if (isGetPass) {
errorTextView.setText("Success!!! Pass = " + result);
} else {
errorTextView.setText("Encrypted pass = " + result);
}
}

@Override
public void onFailure(String message) {
errorTextView.setText("Failed - " + message);
}

@Override
public void onHelp(int helpCode, String helpString) {
errorTextView.setText("Help needed - " + helpString);
}
};
}
}

```

第55.2节：在Android应用中添加指纹扫描器

Android从6.0 (Marshmallow) SDK 23开始支持指纹API

要在您的应用中使用此功能，首先在清单文件中添加 USE_FINGERPRINT 权限。

<uses-permission

Section 55.2: Adding the Fingerprint Scanner in Android application

Android supports fingerprint api from Android 6.0 (Marshmallow) SDK 23

To use this feature in your app, first add the USE_FINGERPRINT permission in your manifest.

<uses-permission

```
        android:name="android.permission.USE_FINGERPRINT" />
```

以下是需要遵循的步骤

首先，您需要使用 KeyGenerator 在 Android 密钥库中创建一个对称密钥，该密钥只能在用户通过指纹认证后使用，并传入一个 KeyGenParameterSpec。

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
    new KeyGenParameterSpec.Builder(KEY_NAME,
        KeyProperties.PURPOSE_SIGN)
    .setDigests(KeyProperties.DIGEST_SHA256)
    .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
    .setUserAuthenticationRequired(true)
.build());
keyPairGenerator.generateKeyPair();
```

通过将 KeyGenParameterSpec.Builder.setUserAuthenticationRequired 设置为 true，您可以允许仅在用户认证后使用该密钥，包括通过用户指纹认证时。

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PublicKey publicKey =
keyStore.getCertificate(MainActivity.KEY_NAME).getPublicKey();

KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PrivateKey key = (PrivateKey) keyStore.getKey(KEY_NAME, null);
```

然后通过调用 FingerprintManager.authenticate 并使用用对称密钥初始化的 Cipher，开始监听指纹传感器上的指纹。或者，您也可以选择回退到服务器端验证的密码作为认证方式。

从 fingerprintManger.class

```
getContext().getSystemService(FingerprintManager.class) 创建并初始化 FingerprintManger
```

要进行认证，使用 FingerprintManger API 并创建子类，继承自

FingerprintManager.AuthenticationCallback 并重写以下方法

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

开始

要开始监听指纹事件，调用带有加密对象的 authenticate 方法

```
指纹管理器
.authenticate(cryptoObject, mCancellationSignal, 0, this, null);
```

```
        android:name="android.permission.USE_FINGERPRINT" />
```

Here the procedure to follow

First you need to create a symmetric key in the Android Key Store using KeyGenerator which can be only be used after the user has authenticated with fingerprint and pass a KeyGenParameterSpec.

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
    new KeyGenParameterSpec.Builder(KEY_NAME,
        KeyProperties.PURPOSE_SIGN)
    .setDigests(KeyProperties.DIGEST_SHA256)
    .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
    .setUserAuthenticationRequired(true)
.build());
keyPairGenerator.generateKeyPair();
```

By setting KeyGenParameterSpec.Builder.setUserAuthenticationRequired to true, you can permit the use of the key only after the user authenticate it including when authenticated with the user's fingerprint.

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PublicKey publicKey =
keyStore.getCertificate(MainActivity.KEY_NAME).getPublicKey();

KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PrivateKey key = (PrivateKey) keyStore.getKey(KEY_NAME, null);
```

Then start listening to a fingerprint on the fingerprint sensor by calling FingerprintManager.authenticate with a Cipher initialized with the symmetric key created. Or alternatively you can fall back to server-side verified password as an authenticator.

Create and initialise the FingerprintManger from fingerprintManger.class

```
getContext().getSystemService(FingerprintManager.class)
```

To authenticate use FingerprintManger api and create subclass using

FingerprintManager.AuthenticationCallback and override the methods

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

To Start

To startListening the fingerPrint event call authenticate method with crypto

```
fingerprintManager
.authenticate(cryptoObject, mCancellationSignal, 0, this, null);
```

取消

停止监听扫描仪调用

```
android.os.CancellationSignal;
```

一旦指纹（或密码）被验证，
FingerprintManager.AuthenticationCallback#onAuthenticationSucceeded() 回调将被调用。

```
@Override
```

```
public void onAuthenticationSucceeded(AuthenticationResult result) {  
}
```

Cancel

to stop listening the scanner call

```
android.os.CancellationSignal;
```

Once the fingerprint (or password) is verified, the
FingerprintManager.AuthenticationCallback#onAuthenticationSucceeded() callback is called.

```
@Override
```

```
public void onAuthenticationSucceeded(AuthenticationResult result) {  
}
```

第56章：蓝牙和蓝牙低功耗 (Bluetooth LE) API

第56.1节：权限

在清单文件中添加此权限以在您的应用中使用蓝牙功能：

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

如果您需要启动设备发现或操作蓝牙设置，还需要添加此权限：

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

针对 Android API 级别 23 及以上版本，需要访问位置信息：

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* 另请参阅权限主题，了解如何正确使用权限的更多详细信息。

第 56.2 节：检查蓝牙是否已启用

```
private static final int REQUEST_ENABLE_BT = 1; // 唯一请求代码
BluetoothAdapter mBluetoothAdapter;

// ...

if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}

// ...

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == RESULT_OK) {
            // 蓝牙已启用
        } else if (resultCode == RESULT_CANCELED) {
            // 蓝牙未启用
        }
    }
}
```

第56.3节：查找附近的蓝牙低功耗设备

BluetoothLE API是在API 18中引入的。然而，扫描设备的方式在API 21中发生了变化。设备搜索必须从定义要扫描的service UUID开始（可以是官方采用的16位UUID或专有UUID）。本示例演示了如何以与API无关的方式搜索BLE设备：

Chapter 56: Bluetooth and Bluetooth LE API

Section 56.1: Permissions

Add this permission to the manifest file to use Bluetooth features in your application:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

If you need to initiate device discovery or manipulate Bluetooth settings, you also need to add this permission:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Targetting Android API level 23 and above, will require location access:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* Also see the Permissions topic for more details on how to use permissions appropriately.

Section 56.2: Check if bluetooth is enabled

```
private static final int REQUEST_ENABLE_BT = 1; // Unique request code
BluetoothAdapter mBluetoothAdapter;

// ...

if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}

// ...

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Bluetooth was enabled
        } else if (resultCode == RESULT_CANCELED) {
            // Bluetooth was not enabled
        }
    }
}
```

Section 56.3: Find nearby Bluetooth Low Energy devices

The BluetoothLE API was introduced in API 18. However, the way of scanning devices has changed in API 21. The searching of devices must start with defining the [service UUID](#) that is to be scanned (either officially adopted 16-bit UUID's or proprietary ones). This example illustrates, how to make an API independent way of searching for BLE devices:

1. 创建蓝牙设备模型：

```
public class BTDevice {  
    String address;  
    String name;  
  
    public String getAddress() {  
        return address;  
    }  
  
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

2. 定义蓝牙扫描接口：

```
public interface ScanningAdapter {  
  
    void startScanning(String name, String[] uuids);  
    void stopScanning();  
    List<BTDevice> getFoundDeviceList();  
}
```

3. 创建扫描工厂类：

```
public class BluetoothScanningFactory implements ScanningAdapter {  
  
    private ScanningAdapter mScanningAdapter;  
  
    public BluetoothScanningFactory() {  
        if (isNewerAPI()) {  
            mScanningAdapter = new LollipopBluetoothLEScanAdapter();  
        } else {  
            mScanningAdapter = new JellyBeanBluetoothLEScanAdapter();  
        }  
    }  
  
    private boolean isNewerAPI() {  
        return Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP;  
    }  
  
    @Override  
    public void startScanning(String[] uuids) {  
        mScanningAdapter.startScanning(uuids);  
    }  
  
    @Override  
    public void stopScanning() {  
        mScanningAdapter.stopScanning();  
    }  
  
    @Override  
    public List<BTDevice> getFoundDeviceList() {
```

1. Create bluetooth device model:

```
public class BTDevice {  
    String address;  
    String name;  
  
    public String getAddress() {  
        return address;  
    }  
  
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

2. Define Bluetooth Scanning interface:

```
public interface ScanningAdapter {  
  
    void startScanning(String name, String[] uuids);  
    void stopScanning();  
    List<BTDevice> getFoundDeviceList();  
}
```

3. Create scanning factory class:

```
public class BluetoothScanningFactory implements ScanningAdapter {  
  
    private ScanningAdapter mScanningAdapter;  
  
    public BluetoothScanningFactory() {  
        if (isNewerAPI()) {  
            mScanningAdapter = new LollipopBluetoothLEScanAdapter();  
        } else {  
            mScanningAdapter = new JellyBeanBluetoothLEScanAdapter();  
        }  
    }  
  
    private boolean isNewerAPI() {  
        return Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP;  
    }  
  
    @Override  
    public void startScanning(String[] uuids) {  
        mScanningAdapter.startScanning(uuids);  
    }  
  
    @Override  
    public void stopScanning() {  
        mScanningAdapter.stopScanning();  
    }  
  
    @Override  
    public List<BTDevice> getFoundDeviceList() {
```

```

        return mScanningAdapter.getFoundDeviceList();
    }
}

```

4. 为每个API创建工厂实现：

API 18 :

```

import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Build;
import android.os.Parcelable;
import android.util.Log;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR2)
public class JellyBeanBluetoothLEScanAdapter implements ScanningAdapter{
    BluetoothAdapter bluetoothAdapter;
    ScanCallback mCallback;
    List<BTDevice> mBluetoothDeviceList;

    public JellyBeanBluetoothLEScanAdapter() {
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        UUID[] uuidList = createUUIDList(uuids);
        bluetoothAdapter.startLeScan(uuidList, mCallback);
    }

    private UUID[] createUUIDList(String[] uuids) {
        UUID[] uuidList = new UUID[uuids.length];
        for (int i = 0 ; i < uuids.length ; ++i) {
            String uuid = uuids[i];
            if (uuid == null) {
                continue;
            }
            uuidList[i] = UUID.fromString(uuid);
        }
        return uuidList;
    }

    @Override
    public void stopScanning() {
        bluetoothAdapter.stopLeScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {

```

```

        return mScanningAdapter.getFoundDeviceList();
    }
}

```

4. Create factory implementation for each API:

API 18:

```

import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Build;
import android.os.Parcelable;
import android.util.Log;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR2)
public class JellyBeanBluetoothLEScanAdapter implements ScanningAdapter{
    BluetoothAdapter bluetoothAdapter;
    ScanCallback mCallback;
    List<BTDevice> mBluetoothDeviceList;

    public JellyBeanBluetoothLEScanAdapter() {
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        UUID[] uuidList = createUUIDList(uuids);
        bluetoothAdapter.startLeScan(uuidList, mCallback);
    }

    private UUID[] createUUIDList(String[] uuids) {
        UUID[] uuidList = new UUID[uuids.length];
        for (int i = 0 ; i < uuids.length ; ++i) {
            String uuid = uuids[i];
            if (uuid == null) {
                continue;
            }
            uuidList[i] = UUID.fromString(uuid);
        }
        return uuidList;
    }

    @Override
    public void stopScanning() {
        bluetoothAdapter.stopLeScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {

```

```

        return mBluetoothDeviceList;
    }

private class ScanCallback implements BluetoothAdapter.LeScanCallback {

    @Override
    public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
        if (isAlreadyAdded(device)) {
            return;
        }
        BTDevice btDevice = new BTDevice();
        btDevice.setName(new String(device.getName()));
        btDevice.setAddress(device.getAddress());
        mBluetoothDeviceList.add(btDevice);
        Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress());
        Parcelable[] uuids = device.getUuids();
        String uuid = "";
        if (uuids != null) {
            for (Parcelable ep : uuids) {
                uuid += ep + " ";
            }
        }
        Log.d("蓝牙发现", device.getName() + " " + device.getAddress() + " " +
        uuid);
    }
}

private boolean isAlreadyAdded(BluetoothDevice bluetoothDevice) {
    for (BTDevice device : mBluetoothDeviceList) {
        String alreadyAddedDeviceMACAddress = device.getAddress();
        String newDeviceMACAddress = bluetoothDevice.getAddress();
        if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
            return true;
        }
    }
    return false;
}
}

```

API 21 :

```

import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.os.Build;
import android.os.ParcelUuid;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class LollipopBluetoothLEScanAdapter implements ScanningAdapter {
    BluetoothLeScanner bluetoothLeScanner;
    ScanCallback mCallback;
    List<BTDevice> mBluetoothDeviceList;

```

```

        return mBluetoothDeviceList;
    }

private class ScanCallback implements BluetoothAdapter.LeScanCallback {

    @Override
    public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
        if (isAlreadyAdded(device)) {
            return;
        }
        BTDevice btDevice = new BTDevice();
        btDevice.setName(new String(device.getName()));
        btDevice.setAddress(device.getAddress());
        mBluetoothDeviceList.add(btDevice);
        Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress());
        Parcelable[] uuids = device.getUuids();
        String uuid = "";
        if (uuids != null) {
            for (Parcelable ep : uuids) {
                uuid += ep + " ";
            }
        }
        Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress() + " " +
        uuid);
    }
}

private boolean isAlreadyAdded(BluetoothDevice bluetoothDevice) {
    for (BTDevice device : mBluetoothDeviceList) {
        String alreadyAddedDeviceMACAddress = device.getAddress();
        String newDeviceMACAddress = bluetoothDevice.getAddress();
        if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
            return true;
        }
    }
    return false;
}
}

```

API 21:

```

import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.os.Build;
import android.os.ParcelUuid;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class LollipopBluetoothLEScanAdapter implements ScanningAdapter {
    BluetoothLeScanner bluetoothLeScanner;
    ScanCallback mCallback;
    List<BTDevice> mBluetoothDeviceList;

```

```

public LollipopBluetoothLEScanAdapter() {
    bluetoothLeScanner = BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
    mCallback = new ScanCallback();
    mBluetoothDeviceList = new ArrayList<>();
}

@Override
public void startScanning(String[] uuids) {
    if (uuids == null || uuids.length == 0) {
        return;
    }
    List<ScanFilter> filterList = createScanFilterList(uuids);
    ScanSettings scanSettings = createScanSettings();
    bluetoothLeScanner.startScan(filterList, scanSettings, mCallback);
}

private List<ScanFilter> createScanFilterList(String[] uuids) {
    List<ScanFilter> filterList = new ArrayList<>();
    for (String uuid : uuids) {
        ScanFilter filter = new ScanFilter.Builder()
            .setServiceUuid(ParcelUuid.fromString(uuid))
            .build();
        filterList.add(filter);
    }
    return filterList;
}

private ScanSettings createScanSettings() {
    ScanSettings settings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_BALANCED)
        .build();
    return settings;
}

@Override
public void stopScanning() {
    bluetoothLeScanner.stopScan(mCallback);
}

@Override
public List<BTDevice> getFoundDeviceList() {
    return mBluetoothDeviceList;
}

public class ScanCallback extends android.bluetooth.le.ScanCallback {

    @Override
    public void onScanResult(int callbackType, ScanResult result) {
        super.onScanResult(callbackType, result);
        if (result == null) {
            return;
        }
        BTDevice device = new BTDevice();
        device.setAddress(result.getDevice().getAddress());
        device.setName(new StringBuffer(result.getScanRecord().getDeviceName()).toString());
        if (device == null || device.getAddress() == null) {
            return;
        }
        if (isAlreadyAdded(device)) {
            return;
        }
        mBluetoothDeviceList.add(device);
    }
}

```

```

public LollipopBluetoothLEScanAdapter() {
    bluetoothLeScanner = BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
    mCallback = new ScanCallback();
    mBluetoothDeviceList = new ArrayList<>();
}

@Override
public void startScanning(String[] uuids) {
    if (uuids == null || uuids.length == 0) {
        return;
    }
    List<ScanFilter> filterList = createScanFilterList(uuids);
    ScanSettings scanSettings = createScanSettings();
    bluetoothLeScanner.startScan(filterList, scanSettings, mCallback);
}

private List<ScanFilter> createScanFilterList(String[] uuids) {
    List<ScanFilter> filterList = new ArrayList<>();
    for (String uuid : uuids) {
        ScanFilter filter = new ScanFilter.Builder()
            .setServiceUuid(ParcelUuid.fromString(uuid))
            .build();
        filterList.add(filter);
    }
    return filterList;
}

private ScanSettings createScanSettings() {
    ScanSettings settings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_BALANCED)
        .build();
    return settings;
}

@Override
public void stopScanning() {
    bluetoothLeScanner.stopScan(mCallback);
}

@Override
public List<BTDevice> getFoundDeviceList() {
    return mBluetoothDeviceList;
}

public class ScanCallback extends android.bluetooth.le.ScanCallback {

    @Override
    public void onScanResult(int callbackType, ScanResult result) {
        super.onScanResult(callbackType, result);
        if (result == null) {
            return;
        }
        BTDevice device = new BTDevice();
        device.setAddress(result.getDevice().getAddress());
        device.setName(new StringBuffer(result.getScanRecord().getDeviceName()).toString());
        if (device == null || device.getAddress() == null) {
            return;
        }
        if (isAlreadyAdded(device)) {
            return;
        }
        mBluetoothDeviceList.add(device);
    }
}

```

```

    }

    private boolean isAlreadyAdded(BTDevice bluetoothDevice) {
        for (BTDevice device : mBluetoothDeviceList) {
            String alreadyAddedDeviceMACAddress = device.getAddress();
            String newDeviceMACAddress = bluetoothDevice.getAddress();
            if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                return true;
            }
        }
        return false;
    }
}

```

5. 通过调用获取已发现设备列表：

```

scanningFactory.startScanning({uuidlist});
等待几秒钟...
List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();

```

第56.4节：使设备可被发现

```

private static final int REQUEST_DISCOVERABLE_BT = 2; // 唯一请求代码
private static final int DISCOVERABLE_DURATION = 120; // 可被发现的持续时间 (秒)
// 0 表示始终可被发现
// 最大值为3600

// ...

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, DISCOVERABLE_DURATION);
startActivityForResult(discoverableIntent, REQUEST_DISCOVERABLE_BT);

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_DISCOVERABLE_BT) {
        if (resultCode == RESULT_OK) {
            // 设备可被发现
        } else if (resultCode == RESULT_CANCELED) {
            // 设备不可被发现
        }
    }
}

```

第56.5节：连接蓝牙设备

在获取BluetoothDevice之后，您可以与其通信。这种通信是通过使用套接字输入\输出流来实现的：

蓝牙通信建立的基本步骤如下：

```

}

private boolean isAlreadyAdded(BTDevice bluetoothDevice) {
    for (BTDevice device : mBluetoothDeviceList) {
        String alreadyAddedDeviceMACAddress = device.getAddress();
        String newDeviceMACAddress = bluetoothDevice.getAddress();
        if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
            return true;
        }
    }
    return false;
}

```

5. Get found device list by calling:

```

scanningFactory.startScanning({uuidlist});
wait few seconds...
List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();

```

Section 56.4: Make device discoverable

```

private static final int REQUEST_DISCOVERABLE_BT = 2; // Unique request code
private static final int DISCOVERABLE_DURATION = 120; // Discoverable duration time in seconds
// 0 means always discoverable
// maximum value is 3600

// ...

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, DISCOVERABLE_DURATION);
startActivityForResult(discoverableIntent, REQUEST_DISCOVERABLE_BT);

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_DISCOVERABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Device is discoverable
        } else if (resultCode == RESULT_CANCELED) {
            // Device is not discoverable
        }
    }
}

```

Section 56.5: Connect to Bluetooth device

After you obtained BluetoothDevice, you can communicate with it. This kind of communication performed by using socket input\output streams:

Those are the basic steps for Bluetooth communication establishment:

1) 初始化套接字：

```
private BluetoothSocket _socket;
//...
public InitializeSocket(BluetoothDevice device){
    try {
        _socket = device.createRfcommSocketToServiceRecord(<您的应用UDID>);
    } catch (IOException e) {
        //错误
    }
}
```

2) 连接套接字：

```
try {
    _socket.connect();
} catch (IOException connEx) {
    try {
        _socket.close();
    } catch (IOException closeException) {
        //错误
    }
}

if (_socket != null && _socket.isConnected()) {
    //Socket 已连接, 现在可以获取我们的 IO 流
}
```

3) 获取套接字输入\输出流

```
private InputStream _inStream;
private OutputStream _outStream;
//...
try {
    _inStream = _socket.getInputStream();
    _outStream = _socket.getOutputStream();
} catch (IOException e) {
    //错误
}
```

输入流 - 用作传入数据通道（接收来自连接设备的数据）
输出流 - 用作传出数据通道

（发送数据到连接设备）完成第三步后，我们可以使用之前初始化的流在两个设备

之间接收和发送数据：

1) 接收数据（从套接字输入流读取）

```
byte[] buffer = new byte[1024]; // 缓冲区（我们的数据）
int bytesCount; // 读取的字节数

while (true) {
    try {
        //从输入流读取数据
        bytesCount = _inStream.read(buffer);
        if(buffer != null && bytesCount > 0)
        {
            //解析接收到的字节
        }
    }
```

1) Initialize socket:

```
private BluetoothSocket _socket;
//...
public InitializeSocket(BluetoothDevice device){
    try {
        _socket = device.createRfcommSocketToServiceRecord(<Your app UDID>);
    } catch (IOException e) {
        //Error
    }
}
```

2) Connect to socket:

```
try {
    _socket.connect();
} catch (IOException connEx) {
    try {
        _socket.close();
    } catch (IOException closeException) {
        //Error
    }
}

if (_socket != null && _socket.isConnected()) {
    //Socket is connected, now we can obtain our IO streams
}
```

3) Obtaining socket Input\Output streams

```
private InputStream _inStream;
private OutputStream _outStream;
//...
try {
    _inStream = _socket.getInputStream();
    _outStream = _socket.getOutputStream();
} catch (IOException e) {
    //Error
}
```

Input stream - Used as incoming data channel (receive data from connected device)

Output stream - Used as outgoing data channel (send data to connected device)

After finishing 3rd step, we can receive and send data between both devices using previously initialized streams:

1) Receiving data (reading from socket input stream)

```
byte[] buffer = new byte[1024]; // buffer (our data)
int bytesCount; // amount of read bytes

while (true) {
    try {
        //reading data from input stream
        bytesCount = _inStream.read(buffer);
        if(buffer != null && bytesCount > 0)
        {
            //Parse received bytes
        }
    }
```

```
        }
    } catch (IOException e) {
        //错误
    }
}
```

2) 发送数据 (写入输出流)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //错误
    }
}
```

- 当然, 连接、读取和写入功能应在专用线程中完成。
- 套接字和流对象需要被

第56.6节：查找附近的蓝牙设备

首先声明一个BluetoothAdapter。

```
BluetoothAdapter mBluetoothAdapter;
```

现在为ACTION_FOUND创建一个BroadcastReceiver

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        //发现设备
        if (BluetoothDevice.ACTION_FOUND.equals(action))
        {
            //从Intent中获取BluetoothDevice对象
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            //将名称和地址添加到数组适配器中以显示在列表中
            mArrayAdapter.add(device.getName() + " " + device.getAddress());
        }
    };
}
```

注册BroadcastReceiver

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

然后通过调用startDiscovery开始发现附近的蓝牙设备

```
mBluetoothAdapter.startDiscovery();
```

别忘了在onDestroy中注销BroadcastReceiver

```
unregisterReceiver(mReceiver);
```

```
        }
    } catch (IOException e) {
        //Error
    }
}
```

2) Sending data (Writing to output stream)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //Error
    }
}
```

- Of course, connection, reading and writing functionality should be done in a dedicated thread.
- Sockets and Stream objects need to be

Section 56.6: Find nearby bluetooth devices

Declare a BluetoothAdapter first.

```
BluetoothAdapter mBluetoothAdapter;
```

Now create a BroadcastReceiver for ACTION_FOUND

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        //Device found
        if (BluetoothDevice.ACTION_FOUND.equals(action))
        {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // Add the name and address to an array adapter to show in a list
            mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    };
};
```

Register the BroadcastReceiver

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Then start discovering the nearby bluetooth devices by calling startDiscovery

```
mBluetoothAdapter.startDiscovery();
```

Don't forget to unregister the BroadcastReceiver inside onDestroy

```
unregisterReceiver(mReceiver);
```

第57章：API-23中的运行时权限

+

Android Marshmallow引入了运行时权限模型。权限被分为两类，即普通权限和危险权限。[危险权限现在由用户在运行时授予。](#)

第57.1节：Android 6.0 多权限

此示例展示了如何在 Android 6 及更高版本中运行时检查权限。

```
public static final int MULTIPLE_PERMISSIONS = 10; // 你想要的代码。

String[] permissions = new String[] {
    Manifest.permission.WRITE_EXTERNAL_STORAGE,
    Manifest.permission.CAMERA,
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.ACCESS_FINE_LOCATION
};

@Override
void onStart() {
    if (checkPermissions()){
        // 权限已授予。
    } else {
        // 显示对话框，告知缺少某些权限
    }
}

private boolean checkPermissions() {
    int result;
    List<String> listPermissionsNeeded = new ArrayList<>();
    for (String p:permissions) {
        result = ContextCompat.checkSelfPermission(getActivity(),p);
        if (result != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p);
        }
    }
    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new String[listPermissionsNeeded.size()]), MULTIPLE_PERMISSIONS);
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MULTIPLE_PERMISSIONS:
            if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                // permissions granted.
            } else {
                // 未授予权限。
            }
            return;
    }
}
```

Chapter 57: Runtime Permissions in API-23

+

Android Marshmallow introduced [Runtime Permission](#) model. Permissions are categorized into two categories i.e. [Normal and Dangerous Permissions](#), where [dangerous permissions](#) are now granted by the user at run time.

Section 57.1: Android 6.0 multiple permissions

This example shows how to check permissions at runtime in Android 6 and later.

```
public static final int MULTIPLE_PERMISSIONS = 10; // code you want.

String[] permissions = new String[] {
    Manifest.permission.WRITE_EXTERNAL_STORAGE,
    Manifest.permission.CAMERA,
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.ACCESS_FINE_LOCATION
};

@Override
void onStart() {
    if (checkPermissions()){
        // permissions granted.
    } else {
        // show dialog informing them that we lack certain permissions
    }
}

private boolean checkPermissions() {
    int result;
    List<String> listPermissionsNeeded = new ArrayList<>();
    for (String p:permissions) {
        result = ContextCompat.checkSelfPermission(getActivity(),p);
        if (result != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p);
        }
    }
    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new String[listPermissionsNeeded.size()]), MULTIPLE_PERMISSIONS);
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MULTIPLE_PERMISSIONS:
            if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                // permissions granted.
            } else {
                // no permissions granted.
            }
            return;
    }
}
```

第57.2节：来自同一权限组的多个运行时权限

在清单文件中，我们有来自两个组的四个危险运行时权限。

```
<!-- 需要读取和写入shredPref文件。 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<!-- 需要获取设备位置。 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

在需要权限的活动中。请注意，重要的是在任何需要权限的活动中检查权限，因为权限可能在应用处于后台时被撤销，应用随后会崩溃。

```
final private int REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS = 124;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.act_layout);

    // 简单检查是否需要管理运行时权限
    if (Build.VERSION.SDK_INT >= 23) {
        checkMultiplePermissions();
    }
}
```

我们只需要从每个权限组中请求其中一个权限，除非用户撤销权限，否则该组中的所有其他权限都被授予。

```
private void checkMultiplePermissions() {

    if (Build.VERSION.SDK_INT >= 23) {
        List<String> permissionsNeeded = new ArrayList<String>();
        List<String> permissionsList = new ArrayList<String>();

        if (!addPermission(permissionsList, android.Manifest.permission.ACCESS_FINE_LOCATION)) {
            permissionsNeeded.add("GPS");
        }

        if (!addPermission(permissionsList, android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
            permissionsNeeded.add("读取存储");
        }

        if (permissionsList.size() > 0) {
            requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;
        }
    }
}
```

```
private boolean addPermission(List<String> permissionsList, String permission) {
    if (Build.VERSION.SDK_INT >= 23)
```

Section 57.2: Multiple Runtime Permissions From Same Permission Groups

In the manifest we have fours dangerous runtime permissions from two groups.

```
<!-- Required to read and write to shredPref file. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<!-- Required to get location of device. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

In the activity where the permissions are required. Note it is important to check for permissions in any activity that requires permissions, as the permissions can be revoked while the app is in the background and the app will then crash.

```
final private int REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS = 124;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.act_layout);

    // A simple check of whether runtime permissions need to be managed
    if (Build.VERSION.SDK_INT >= 23) {
        checkMultiplePermissions();
    }
}
```

We only need to ask for permission for one of these from each group and all other permissions from this group are granted unless the permission is revoked by the user.

```
private void checkMultiplePermissions() {

    if (Build.VERSION.SDK_INT >= 23) {
        List<String> permissionsNeeded = new ArrayList<String>();
        List<String> permissionsList = new ArrayList<String>();

        if (!addPermission(permissionsList, android.Manifest.permission.ACCESS_FINE_LOCATION)) {
            permissionsNeeded.add("GPS");
        }

        if (!addPermission(permissionsList, android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
            permissionsNeeded.add("Read Storage");
        }

        if (permissionsList.size() > 0) {
            requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;
        }
    }
}
```

```
private boolean addPermission(List<String> permissionsList, String permission) {
    if (Build.VERSION.SDK_INT >= 23)
```

```

if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
    permissionsList.add(permission);

    // 检查理由选项
    if (!shouldShowRequestPermissionRationale(permission))
        return false;
}
return true;
}

```

这处理用户是否允许权限的结果。在此示例中，如果权限未被允许，应用程序将被终止。

```

@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS: {

            Map<String, Integer> perms = new HashMap<String, Integer>();
            // 初始化
            perms.put(android.Manifest.permission.ACCESS_FINE_LOCATION,
                    PackageManager.PERMISSION_GRANTED);
            perms.put(android.Manifest.permission.READ_EXTERNAL_STORAGE,
                    PackageManager.PERMISSION_GRANTED);

            // 填充结果
            for (int i = 0; i < permissions.length; i++)
                perms.put(permissions[i], grantResults[i]);
            if (perms.get(android.Manifest.permission.ACCESS_FINE_LOCATION) ==
                    PackageManager.PERMISSION_GRANTED
                    && perms.get(android.Manifest.permission.READ_EXTERNAL_STORAGE) ==
                    PackageManager.PERMISSION_GRANTED) {
                // 所有权限已授予
                return;
            } else {
                // 权限被拒绝
                if (Build.VERSION.SDK_INT >= 23) {
                    Toast.makeText(
                            getApplicationContext(),
                            "我的应用无法在没有定位和存储" +
                            "权限的情况下运行。
                            请重新启动我的应用或在应用设置中允许权限",
                            Toast.LENGTH_LONG).show();
                    finish();
                }
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}

```

更多信息

<https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en>

第57.3节：使用PermissionUtil

PermissionUtil是一种简单便捷的上下文权限请求方式。你可以轻松地提供在所有请求权限被授予时应执行的操作（onAllGranted()）、任何请求被拒绝时的操作

```

if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
    permissionsList.add(permission);

    // Check for Rationale Option
    if (!shouldShowRequestPermissionRationale(permission))
        return false;
}
return true;
}

```

This deals with the result of the user allowing or not allowing permissions. In this example, if the permissions are not allowed, the app is killed.

```

@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS: {

            Map<String, Integer> perms = new HashMap<String, Integer>();
            // Initial
            perms.put(android.Manifest.permission.ACCESS_FINE_LOCATION,
                    PackageManager.PERMISSION_GRANTED);
            perms.put(android.Manifest.permission.READ_EXTERNAL_STORAGE,
                    PackageManager.PERMISSION_GRANTED);

            // Fill with results
            for (int i = 0; i < permissions.length; i++)
                perms.put(permissions[i], grantResults[i]);
            if (perms.get(android.Manifest.permission.ACCESS_FINE_LOCATION) ==
                    PackageManager.PERMISSION_GRANTED
                    && perms.get(android.Manifest.permission.READ_EXTERNAL_STORAGE) ==
                    PackageManager.PERMISSION_GRANTED) {
                // All Permissions Granted
                return;
            } else {
                // Permission Denied
                if (Build.VERSION.SDK_INT >= 23) {
                    Toast.makeText(
                            getApplicationContext(),
                            "My App cannot run without Location and Storage " +
                            "Permissions.\nRelaunch My App or allow permissions" +
                            " in Applications Settings",
                            Toast.LENGTH_LONG).show();
                    finish();
                }
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}

```

More Information

<https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en>

Section 57.3: Using PermissionUtil

PermissionUtil is a simple and convenient way of asking for permissions in context. You can easily provide what should happen in case of all requested permissions granted (onAllGranted()), any request was denied

(onAnyDenied()) 或需要说明理由时的操作 (onRational())。

在你的AppCompatActivity或Fragment中任何需要请求用户权限的地方

```
mRequestObject =  
PermissionUtil.with(this).request(Manifest.permission.WRITE_EXTERNAL_STORAGE).onAllGranted(  
    new Func() {  
        @Override protected void call() {  
            //正常路径  
        }  
    }).onAnyDenied(  
    new Func() {  
        @Override protected void call() {  
            //错误路径  
        }  
    }).ask(REQUEST_CODE_STORAGE);
```

并将此添加到 onRequestPermissionsResult

```
if(mRequestObject!=null){  
mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);  
}
```

还需将请求的权限添加到您的 AndroidManifest.xml 中

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

第57.4节：将所有权限相关代码包含到一个抽象基类中，并继承该基类的活动以实现更清晰/可重用的代码

```
public abstract class BaseActivity extends AppCompatActivity {  
    private Map<Integer, PermissionCallback> permissionCallbackMap = new HashMap<>();  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
        ...  
    }  
  
    @Override  
    public void setContentView(int layoutResId) {  
        super.setContentView(layoutResId);  
        bindViews();  
    }  
  
    ...  
  
    @Override  
    public void onRequestPermissionsResult(  
        int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
        PermissionCallback callback = permissionCallbackMap.get(requestCode);  
  
        if (callback == null) return;  
  
        // 检查权限请求是否被拒绝。  
        if (grantResults.length < 0 && permissions.length > 0) {  
            callback.onPermissionDenied(permissions);  
        }  
    }
```

(onAnyDenied()) or in case that a rational is needed (onRational()).

Anywhere in your AppCompatActivity or Fragment that you want to ask for user's permission

```
mRequestObject =  
PermissionUtil.with(this).request(Manifest.permission.WRITE_EXTERNAL_STORAGE).onAllGranted(  
    new Func() {  
        @Override protected void call() {  
            //Happy Path  
        }  
    }).onAnyDenied(  
    new Func() {  
        @Override protected void call() {  
            //Sad Path  
        }  
    }).ask(REQUEST_CODE_STORAGE);
```

And add this to onRequestPermissionsResult

```
if(mRequestObject!=null){  
mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);  
}
```

Add the requested permission to your AndroidManifest.xml as well

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Section 57.4: Include all permission-related code to an abstract base class and extend the activity of this base class to achieve cleaner/reusable code

```
public abstract class BaseActivity extends AppCompatActivity {  
    private Map<Integer, PermissionCallback> permissionCallbackMap = new HashMap<>();  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
        ...  
    }  
  
    @Override  
    public void setContentView(int layoutResId) {  
        super.setContentView(layoutResId);  
        bindViews();  
    }  
  
    ...  
  
    @Override  
    public void onRequestPermissionsResult(  
        int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
        PermissionCallback callback = permissionCallbackMap.get(requestCode);  
  
        if (callback == null) return;  
  
        // Check whether the permission request was rejected.  
        if (grantResults.length < 0 && permissions.length > 0) {  
            callback.onPermissionDenied(permissions);  
        }  
    }
```

```

        return;
    }

List<String> grantedPermissions = new ArrayList<>();
List<String> blockedPermissions = new ArrayList<>();
List<String> deniedPermissions = new ArrayList<>();
int index = 0;

for (String permission : permissions) {
List<String> permissionList = grantResults[index] == PackageManager.PERMISSION_GRANTED
    ? grantedPermissions
    : ! ActivityCompat.shouldShowRequestPermissionRationale(this, permission)
        ? blockedPermissions
        : deniedPermissions;
    permissionList.add(permission);
    index++;
}

if (grantedPermissions.size() > 0) {
callback.onPermissionGranted(
grantedPermissions.toArray(new String[grantedPermissions.size()]));
}

if (deniedPermissions.size() > 0) {
callback.onPermissionDenied(
deniedPermissions.toArray(new String[deniedPermissions.size()]));
}

if (blockedPermissions.size() > 0) {
callback.onPermissionBlocked(
blockedPermissions.toArray(new String[blockedPermissions.size()]));

}

permissionCallbackMap.remove(requestCode);
}

/**
 * 检查权限是否被授予。
 *
 * @param permission
 * @return
 */
public boolean hasPermission(String permission) {
    return ContextCompat.checkSelfPermission(this, permission) ==
PackageManager.PERMISSION_GRANTED;
}

/**
 * 请求权限并在回调中获取结果。
 *
 * @param permissions
 * @param callback
 */
public void requestPermission(String [] permissions, @NonNull PermissionCallback callback) {
    int requestCode = permissionCallbackMap.size() + 1;
    permissionCallbackMap.put(requestCode, callback);
    ActivityCompat.requestPermissions(this, permissions, requestCode);
}

/**
 * 请求权限并在回调中获取结果。
 *
 */

```

```

        return;
    }

List<String> grantedPermissions = new ArrayList<>();
List<String> blockedPermissions = new ArrayList<>();
List<String> deniedPermissions = new ArrayList<>();
int index = 0;

for (String permission : permissions) {
List<String> permissionList = grantResults[index] == PackageManager.PERMISSION_GRANTED
    ? grantedPermissions
    : ! ActivityCompat.shouldShowRequestPermissionRationale(this, permission)
        ? blockedPermissions
        : deniedPermissions;
    permissionList.add(permission);
    index++;
}

if (grantedPermissions.size() > 0) {
callback.onPermissionGranted(
grantedPermissions.toArray(new String[grantedPermissions.size()]));
}

if (deniedPermissions.size() > 0) {
callback.onPermissionDenied(
deniedPermissions.toArray(new String[deniedPermissions.size()]));
}

if (blockedPermissions.size() > 0) {
callback.onPermissionBlocked(
blockedPermissions.toArray(new String[blockedPermissions.size()]));

}

permissionCallbackMap.remove(requestCode);
}

/**
 * Check whether a permission is granted or not.
 *
 * @param permission
 * @return
 */
public boolean hasPermission(String permission) {
    return ContextCompat.checkSelfPermission(this, permission) ==
PackageManager.PERMISSION_GRANTED;
}

/**
 * Request permissions and get the result on callback.
 *
 * @param permissions
 * @param callback
 */
public void requestPermission(String [] permissions, @NonNull PermissionCallback callback) {
    int requestCode = permissionCallbackMap.size() + 1;
    permissionCallbackMap.put(requestCode, callback);
    ActivityCompat.requestPermissions(this, permissions, requestCode);
}

/**
 * Request permission and get the result on callback.
 *
 */

```

```

* @param permission
* @param callback
*/
public void requestPermission(String permission, @NonNull PermissionCallback callback) {
    int requestCode = permissionCallbackMap.size() + 1;
    permissionCallbackMap.put(requestCode, callback);
    ActivityCompat.requestPermissions(this, new String[] { permission }, requestCode);
}

```

活动中的示例用法

该活动应继承上述定义的抽象基类，如下所示：

```

private void requestLocationAfterPermissionCheck() {
    if (hasPermission(Manifest.permission.ACCESS_FINE_LOCATION)) {
        requestLocation();
        return;
    }

    // 调用基类方法。
requestPermission(Manifest.permission.ACCESS_FINE_LOCATION, new PermissionCallback() {
    @Override
    public void onPermissionGranted(String[] grantedPermissions) {
        requestLocation();
    }

    @Override
    public void onPermissionDenied(String[] deniedPermissions) {
        // 执行某些操作。
    }

    @Override
    public void onPermissionBlocked(String[] blockedPermissions) {
        // 执行某些操作。
    }
});
}

```

第57.5节：在广播和URI中强制执行权限

您可以在向已注册的广播接收器发送Intent时进行权限检查。您发送的权限会与标签下注册的权限进行交叉核对。它们限制了谁可以向相关接收器发送广播。

要发送带权限的广播请求，请在

`Context.sendBroadcast(Intent intent, String permission)` 调用中指定权限字符串，但请注意，接收者的应用必须拥有该权限才能接收您的广播。接收器应先于发送者安装。

方法名为：

```

void sendBroadcast (Intent intent, String receiverPermission)
//例如，向Bcastreceiver接收器发送广播
Intent broadcast = new Intent(this, Bcastreceiver.class);
sendBroadcast(broadcast, "org.quadcore.mypermission");

```

你可以在清单文件中指定广播发送者必须包含通过`sendBroadcast`发送的请求权限：

```

* @param permission
* @param callback
*/
public void requestPermission(String permission, @NonNull PermissionCallback callback) {
    int requestCode = permissionCallbackMap.size() + 1;
    permissionCallbackMap.put(requestCode, callback);
    ActivityCompat.requestPermissions(this, new String[] { permission }, requestCode);
}

```

Example usage in the activity

The activity should extend the abstract base class defined above as follows:

```

private void requestLocationAfterPermissionCheck() {
    if (hasPermission(Manifest.permission.ACCESS_FINE_LOCATION)) {
        requestLocation();
        return;
    }

    // Call the base class method.
requestPermission(Manifest.permission.ACCESS_FINE_LOCATION, new PermissionCallback() {
    @Override
    public void onPermissionGranted(String[] grantedPermissions) {
        requestLocation();
    }

    @Override
    public void onPermissionDenied(String[] deniedPermissions) {
        // Do something.
    }

    @Override
    public void onPermissionBlocked(String[] blockedPermissions) {
        // Do something.
    }
});
}

```

Section 57.5: Enforcing Permissions in Broadcasts, URI

You can do a permissions check when sending an Intent to a registered broadcast receiver. The permissions you send are cross-checked with the ones registered under the tag. They restrict who can send broadcasts to the associated receiver.

To send a broadcast request with permissions, specify the permission as a string in the `Context.sendBroadcast(Intent intent, String permission)` call, but keep in mind that the Receiver's app **MUST** have that permission in order to receive your broadcast. The receiver should be installed first before the sender.

The method signature is:

```

void sendBroadcast (Intent intent, String receiverPermission)
//for example to send a broadcast to Bcastreceiver receiver
Intent broadcast = new Intent(this, Bcastreceiver.class);
sendBroadcast(broadcast, "org.quadcore.mypermission");

```

and you can specify in your manifest that the broadcast sender is required to include the requested permission sent through the `sendBroadcast`:

```
<!-- 你的特殊权限 -->
<permission android:name="org.quadcore.mypermission"
    android:label="my_permission"
    android:protectionLevel="dangerous"></permission>
```

还需在预期接收此广播的应用的清单文件中声明该权限：

```
<!-- 我使用该权限！ -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- 以及接收器 -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

注意：接收器和广播器都可以要求权限，当发生这种情况时，必须通过两个权限检查才能将Intent传递给相关目标。定义该权限的应用应先安装。

在此处查找完整的权限文档。 [here](#)

```
<!-- Your special permission -->
<permission android:name="org.quadcore.mypermission"
    android:label="my_permission"
    android:protectionLevel="dangerous"></permission>
```

Also declare the permission in the manifest of the application that is supposed to receive this broadcast:

```
<!-- I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Note: Both a receiver and a broadcaster can require a permission, and when this happens, both permission checks must pass for the Intent to be delivered to the associated target. The App that defines the permission should be installed first.

Find the full documentation [here](#) on Permissions.

第58章：Android 地点API

第58.1节：使用地点API获取当前位置

您可以使用Google Places API获取用户的当前位置和本地地点。

首先，您应调用`PlaceDetectionApi.getCurrentPlace()`方法以检索本地商家或其他地点。该方法返回一个`PlaceLikelihoodBuffer`对象，其中包含`PlaceLikelihood`对象列表。然后，您可以通过调用`PlaceLikelihood.getPlace()`方法获取一个`Place`对象。

重要提示：您必须请求并获得`ACCESS_FINE_LOCATION`权限，才能允许您的应用访问精确的位置信息。

```
private static final int PERMISSION_REQUEST_TO_ACCESS_LOCATION = 1;

private TextView txtLocation;
private GoogleApiClient googleApiClient;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_location);

    txtLocation = (TextView) this.findViewById(R.id.txtLocation);
    googleApiClient = new GoogleApiClient.Builder(this)
        .addApi(Places.GEO_DATA_API)
        .addApi(Places.PLACE_DETECTION_API)
        .enableAutoManage(this, this)
        .build();

    getCurrentLocation();
}

private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        Log.e(LOG_TAG, "Permission is not granted");

        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSION_REQUEST_TO_ACCESS_LOCATION);
        return;
    }

    Log.i(LOG_TAG, "Permission is granted");

    PendingResult<PlaceLikelihoodBuffer> result =
    Places.PlaceDetectionApi.getCurrentPlace(googleApiClient, null);
    result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>() {
        @Override
        public void onResult(PlaceLikelihoodBuffer likelyPlaces) {
            Log.i(LOG_TAG, String.format("Result received : %d ", likelyPlaces.getCount()));
            StringBuilder stringBuilder = new StringBuilder();

            for (PlaceLikelihood placeLikelihood : likelyPlaces) {
                stringBuilder.append(String.format("Place : '%s' %n",
                    placeLikelihood.getPlace().getName()));
            }
            likelyPlaces.release();
            txtLocation.setText(stringBuilder.toString());
        }
    });
}
```

Chapter 58: Android Places API

Section 58.1: Getting Current Places by Using Places API

You can get the current location and local places of user by using the [Google Places API](#).

Ar first, you should call the `PlaceDetectionApi.getCurrentPlace()` method in order to retrieve local business or other places. This method returns a `PlaceLikelihoodBuffer` object which contains a list of `PlaceLikelihood` objects. Then, you can get a `Place` object by calling the `PlaceLikelihood.getPlace()` method.

Important: You must request and obtain the `ACCESS_FINE_LOCATION` permission in order to allow your app to access precise location information.

```
private static final int PERMISSION_REQUEST_TO_ACCESS_LOCATION = 1;

private TextView txtLocation;
private GoogleApiClient googleApiClient;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_location);

    txtLocation = (TextView) this.findViewById(R.id.txtLocation);
    googleApiClient = new GoogleApiClient.Builder(this)
        .addApi(Places.GEO_DATA_API)
        .addApi(Places.PLACE_DETECTION_API)
        .enableAutoManage(this, this)
        .build();

    getCurrentLocation();
}

private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        Log.e(LOG_TAG, "Permission is not granted");

        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSION_REQUEST_TO_ACCESS_LOCATION);
        return;
    }

    Log.i(LOG_TAG, "Permission is granted");

    PendingResult<PlaceLikelihoodBuffer> result =
    Places.PlaceDetectionApi.getCurrentPlace(googleApiClient, null);
    result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>() {
        @Override
        public void onResult(PlaceLikelihoodBuffer likelyPlaces) {
            Log.i(LOG_TAG, String.format("Result received : %d ", likelyPlaces.getCount()));
            StringBuilder stringBuilder = new StringBuilder();

            for (PlaceLikelihood placeLikelihood : likelyPlaces) {
                stringBuilder.append(String.format("Place : '%s' %n",
                    placeLikelihood.getPlace().getName()));
            }
            likelyPlaces.release();
            txtLocation.setText(stringBuilder.toString());
        }
    });
}
```

```

        });

    }

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case PERMISSION_REQUEST_TO_ACCESS_LOCATION: {
            // 如果请求被取消，结果数组将为空。
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                getCurrentLocation();
            } else {
                // 权限被拒绝，哎！
                // 禁用依赖此权限的功能。
            }
            return;
        }
        // 添加更多的'case'分支以检查此应用可能请求的其他权限。
    }
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Log.e(LOG_TAG, "GoogleApiClient 连接失败: " + connectionResult.getErrorMessage());
}

```

第58.2节：地点自动完成集成

Google Places API for Android 中的自动完成功能为用户提供地点预测。当用户在搜索框中输入时，自动完成会根据用户的查询显示地点。

AutoCompleteActivity.java

```

private TextView txtSelectedPlaceName;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_autocomplete);

    txtSelectedPlaceName = (TextView) this.findViewById(R.id.txtSelectedPlaceName);

    PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
        getFragmentManager().findFragmentById(R.id.fragment_autocomplete);

    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            Log.i(LOG_TAG, "地点: " + place.getName());
            txtSelectedPlaceName.setText(String.format("选定地点 : %s - %s",
                place.getName(), place.getAddress()));
        }

        @Override
        public void onError(Status status) {
            Log.i(LOG_TAG, "发生错误: " + status);
            Toast.makeText(AutoCompleteActivity.this, "无法选择地点!!",
                Toast.LENGTH_SHORT).show();
        }
    });
}

```

```

        });

    }

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case PERMISSION_REQUEST_TO_ACCESS_LOCATION: {
            // If the request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                getCurrentLocation();
            } else {
                // Permission denied, boo!
                // Disable the functionality that depends on this permission.
            }
            return;
        }
        // Add further 'case' lines to check for other permissions this app might request.
    }
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Log.e(LOG_TAG, "GoogleApiClient connection failed: " + connectionResult.getErrorMessage());
}

```

Section 58.2: Place Autocomplete Integration

The autocomplete feature in the Google Places API for Android provides place predictions to user. While user types in the search box, autocomplete shows places according to user's queries.

AutoCompleteActivity.java

```

private TextView txtSelectedPlaceName;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_autocomplete);

    txtSelectedPlaceName = (TextView) this.findViewById(R.id.txtSelectedPlaceName);

    PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
        getFragmentManager().findFragmentById(R.id.fragment_autocomplete);

    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            Log.i(LOG_TAG, "Place: " + place.getName());
            txtSelectedPlaceName.setText(String.format("Selected places : %s - %s",
                place.getName(), place.getAddress()));
        }

        @Override
        public void onError(Status status) {
            Log.i(LOG_TAG, "An error occurred: " + status);
            Toast.makeText(AutoCompleteActivity.this, "Place cannot be selected!!",
                Toast.LENGTH_SHORT).show();
        }
    });
}

```

```
});  
}  
  
activity_autocomplete.xml
```

```
<fragment  
    android:id="@+id/fragment_autocomplete"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"  
/>
```

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/txtSelectedPlaceName"  
    android:layout_margin="20dp"  
    android:padding="15dp"  
    android:hint="@string/txt_select_place_hint"  
    android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
```

第58.3节：地点选择器使用示例

地点选择器是由地点API提供的一个非常简单的用户界面控件。它提供了内置地图、当前位置、附近地点、搜索功能和自动完成功能。

这是一个地点选择器UI控件的示例用法。

```
private static int PLACE_PICKER_REQUEST = 1;  
  
private TextView txtPlaceName;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_place_picker_sample);  
  
    txtPlaceName = (TextView) this.findViewById(R.id.txtPlaceName);  
    Button btnSelectPlace = (Button) this.findViewById(R.id.btnSelectPlace);  
    btnSelectPlace.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            openPlacePickerView();  
        }  
    });  
  
    private void openPlacePickerView(){  
        PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();  
        try {  
            startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);  
        } catch (GooglePlayServicesRepairableException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
});  
}  
  
activity_autocomplete.xml
```

```
<fragment  
    android:id="@+id/fragment_autocomplete"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"  
/>
```

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/txtSelectedPlaceName"  
    android:layout_margin="20dp"  
    android:padding="15dp"  
    android:hint="@string/txt_select_place_hint"  
    android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
```

Section 58.3: Place Picker Usage Example

Place Picker is a really simple UI widget provided by Places API. It provides a built-in map, current location, nearby places, search abilities and autocomplete.

This is a sample usage of Place Picker UI widget.

```
private static int PLACE_PICKER_REQUEST = 1;  
  
private TextView txtPlaceName;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_place_picker_sample);  
  
    txtPlaceName = (TextView) this.findViewById(R.id.txtPlaceName);  
    Button btnSelectPlace = (Button) this.findViewById(R.id.btnSelectPlace);  
    btnSelectPlace.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            openPlacePickerView();  
        }  
    });  
  
    private void openPlacePickerView(){  
        PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();  
        try {  
            startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);  
        } catch (GooglePlayServicesRepairableException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

        } catch (GooglePlayServicesNotAvailableException e) {
e.printStackTrace();
    }

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
Place place = PlacePicker.getPlace(this, data);
        Log.i(LOG_TAG, String.format("地点名称 : %s", place.getName()));
        Log.i(LOG_TAG, String.format("地点地址 : %s", place.getAddress()));
        Log.i(LOG_TAG, String.format("地点ID : %s", place.getId()));

txtPlaceName.setText(String.format("地点 : %s - %s", place.getName(),
place.getAddress()));
    }
}
}

```

第58.4节：为PlaceAutocomplete设置地点类型过滤器

在某些场景中，我们可能希望将PlaceAutocomplete显示的结果缩小到特定国家，或者只显示区域。这可以通过在意图上设置一个AutocompleteFilter来实现。例如，如果我只想查找类型为REGION且仅属于印度的地点，我会这样做：

MainActivity.java

```

public class MainActivity extends AppComatActivity {

private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
private TextView selectedPlace;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

    selectedPlace = (TextView) findViewById(R.id.selected_place);
    try {
AutocompleteFilter typeFilter = new AutocompleteFilter.Builder()
        .setTypeFilter(AutocompleteFilter.TYPE_FILTER_REGIONS)
        .setCountry("IN")
.build();

    Intent intent =
        new PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
            .setFilter(typeFilter)
            .build(this);
startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);

    } catch (GooglePlayServicesRepairableException
        | GooglePlayServicesNotAvailableException e) {
e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode,
                               int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE && resultCode == Activity.RESULT_OK) {

```

```

        } catch (GooglePlayServicesNotAvailableException e) {
e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
Place place = PlacePicker.getPlace(this, data);
        Log.i(LOG_TAG, String.format("Place Name : %s", place.getName()));
        Log.i(LOG_TAG, String.format("Place Address : %s", place.getAddress()));
        Log.i(LOG_TAG, String.format("Place Id : %s", place.getId()));

txtPlaceName.setText(String.format("Place : %s - %s", place.getName(),
place.getAddress()));
    }
}
}

```

Section 58.4: Setting place type filters for PlaceAutocomplete

In some scenarios, we might want to narrow down the results being shown by **PlaceAutocomplete** to a specific country or maybe to show only Regions. This can be achieved by setting an **AutocompleteFilter** on the intent. For example, if I want to look only for places of type REGION and only belonging to India, I would do the following:

MainActivity.java

```

public class MainActivity extends AppComatActivity {

private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
private TextView selectedPlace;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

    selectedPlace = (TextView) findViewById(R.id.selected_place);
    try {
AutocompleteFilter typeFilter = new AutocompleteFilter.Builder()
        .setTypeFilter(AutocompleteFilter.TYPE_FILTER_REGIONS)
        .setCountry("IN")
.build();

    Intent intent =
        new PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
            .setFilter(typeFilter)
            .build(this);
startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);

    } catch (GooglePlayServicesRepairableException
        | GooglePlayServicesNotAvailableException e) {
e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode,
                               int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE && resultCode == Activity.RESULT_OK) {

```

```

final Place place = PlacePicker.getPlace(this, data);
selectedPlace.setText(place.getName().toString().toUpperCase());
} else {
Toast.makeText(MainActivity.this, "无法获取位置。", Toast.LENGTH_SHORT).show();
}
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/selected_place"/>

</LinearLayout>

```

PlaceAutocomplete将自动启动，然后您可以从结果中选择一个地点，这些结果仅为REGION类型，并且仅属于指定的国家。该意图也可以通过点击按钮启动。

第58.5节：添加多个Google自动完成活动

```

public static final int PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE=1;
public static final int PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE=2;

fromPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        try {
            //处理起点相关操作
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: 处理错误。
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: 处理错误。
        }
    }
});
toPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        try {
            //执行你的操作以放置
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: 处理错误。
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: 处理错误。
        }
    }
});

```

```

final Place place = PlacePicker.getPlace(this, data);
selectedPlace.setText(place.getName().toString().toUpperCase());
} else {
Toast.makeText(MainActivity.this, "Could not get location.", Toast.LENGTH_SHORT).show();
}
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/selected_place"/>

</LinearLayout>

```

The **PlaceAutocomplete** will launch automatically and you can then select a place from the results which will only be of the type **REGION** and will only belong to the specified country. The intent can also be launched at the click of a button.

Section 58.5: Adding more than one google auto complete activity

```

public static final int PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE=1;
public static final int PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE=2;

fromPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        try {
            //Do your stuff from place
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: Handle the error.
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: Handle the error.
        }
    }
});
toPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        try {
            //Do your stuff to place
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: Handle the error.
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: Handle the error.
        }
    }
});

```

```

        }
    });

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            //在这里处理你的“从地点”操作
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            //处理你的“从地点”错误
        } else if (resultCode == RESULT_CANCELED) {
            // 用户取消了操作。
        }
    } else if (requestCode == PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            //在这里处理你的“到地点”操作
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            //处理你的“到地点”错误
        } else if (resultCode == RESULT_CANCELED) {
            // 用户取消了操作。
        }
    }
}

```

```

        }
    });

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            //Do your ok >from place< stuff here
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            //Handle your error >from place<
        } else if (resultCode == RESULT_CANCELED) {
            // The user canceled the operation.
        }
    } else if (requestCode == PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            //Do your ok >to place< stuff here
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            //Handle your error >to place<
        } else if (resultCode == RESULT_CANCELED) {
            // The user canceled the operation.
        }
    }
}

```

第59章：Android NDK

第59.1节：如何在ndk中进行日志记录

首先确保在你的Android.mk文件中链接了日志库：

```
LOCAL_LDLIBS := -llog
```

然后使用以下任一__android_log_print()调用：

```
#include <android/log.h>
#define TAG "MY LOG"

__android_log_print(ANDROID_LOG_VERBOSE, TAG, "1 + 1 的值是 %d", 1 + 1)
__android_log_print(ANDROID_LOG_WARN, TAG, "1 + 1 的值是 %d", 1 + 1)
__android_log_print(ANDROID_LOG_DEBUG, TAG, "1 + 1 的值是 %d", 1 + 1)
__android_log_print(ANDROID_LOG_INFO, TAG, "1 + 1 的值是 %d", 1 + 1)
__android_log_print(ANDROID_LOG_ERROR, TAG, "1 + 1 的值是 %d", 1 + 1)
```

或者通过定义相应的宏以更方便的方式使用它们：

```
#define LOGV(...) __android_log_print(ANDROID_LOG_VERBOSE, TAG, __VA_ARGS__)
#define LOGW(...) __android_log_print(ANDROID_LOG_WARN, TAG, __VA_ARGS__)
#define LOGD(...) __android_log_print(ANDROID_LOG_DEBUG, TAG, __VA_ARGS__)
#define LOGI(...) __android_log_print(ANDROID_LOG_INFO, TAG, __VA_ARGS__)
#define LOGE(...) __android_log_print(ANDROID_LOG_ERROR, TAG, __VA_ARGS__)
```

示例：

```
int x = 42;
LOGD("变量 x 的值是 %d", x);
```

第59.2节：为Android构建本地可执行文件

project/jni/main.c

```
#include <stdio.h>
#include <unistd.h>
int main(void) {
    printf("你好，世界！");
    return 0;
}
```

project/jni/Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE := hello_world
LOCAL_SRC_FILES := main.c
include $(BUILD_EXECUTABLE)
```

project/jni/Application.mk

Chapter 59: Android NDK

Section 59.1: How to log in ndk

First make sure you link against the logging library in your `Android.mk` file:

```
LOCAL_LDLIBS := -llog
```

Then use one of the following `__android_log_print()` calls:

```
#include <android/log.h>
#define TAG "MY LOG"

__android_log_print(ANDROID_LOG_VERBOSE, TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_WARN, TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_DEBUG, TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_INFO, TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_ERROR, TAG, "The value of 1 + 1 is %d", 1 + 1)
```

Or use those in a more convenient way by defining corresponding macros:

```
#define LOGV(...) __android_log_print(ANDROID_LOG_VERBOSE, TAG, __VA_ARGS__)
#define LOGW(...) __android_log_print(ANDROID_LOG_WARN, TAG, __VA_ARGS__)
#define LOGD(...) __android_log_print(ANDROID_LOG_DEBUG, TAG, __VA_ARGS__)
#define LOGI(...) __android_log_print(ANDROID_LOG_INFO, TAG, __VA_ARGS__)
#define LOGE(...) __android_log_print(ANDROID_LOG_ERROR, TAG, __VA_ARGS__)
```

Example:

```
int x = 42;
LOGD("The value of x is %d", x);
```

Section 59.2: Building native executables for Android

project/jni/main.c

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
    printf("Hello world!\n");
    return 0;
}
```

project/jni/Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE := hello_world
LOCAL_SRC_FILES := main.c
include $(BUILD_EXECUTABLE)
```

project/jni/Application.mk

```
APP_ABI := all  
APP_PLATFORM := android-21
```

如果您想支持运行 Android 5.0 (API 21) 以下版本的设备，您需要将二进制文件编译时的APP_PLATFORM设置为较旧的 API，例如android-8。这是因为 Android 5.0 强制执行位置无关二进制文件 (PIE)，而较旧的设备不一定支持 PIE。因此，您需要根据设备版本使用 PIE 或非 PIE。如果您想在 Android 应用程序中使用该二进制文件，您需要检查 API 级别并提取正确的二进制文件。

APP_ABI 可以更改为特定平台，例如 armeabi，以仅为这些架构构建二进制文件。

在最坏的情况下，您将为每种架构拥有一个PIE和一个非PIE二进制文件（使用ndk-r10e时约有14个不同的二进制文件）。

构建可执行文件：

```
cd project  
ndk-build
```

您可以在project/libs/<architecture>/hello_world找到二进制文件。您可以通过ADB (push并 chmod赋予可执行权限) 使用它，或者从您的应用程序中提取并chmod赋予可执行权限后使用。

要确定CPU的架构，请获取主架构的构建属性 ro.product.cpu.abi，或在较新设备上获取支持架构的完整列表 r o.product.cpu.abi.list。您可以在应用程序中使用 android.os.Build 类，或通过ADB使用 getprop <name> 来完成此操作。

第59.3节：如何清理构建

如果您需要清理构建：

```
ndk-build clean
```

第59.4节：如何使用除Android.mk以外的makefile

```
ndk-build NDK_PROJECT_PATH=PROJECT_PATH APP_BUILD_SCRIPT=MyAndroid.mk
```

```
APP_ABI := all  
APP_PLATFORM := android-21
```

If you want to support devices running Android versions lower than 5.0 (API 21), you need to compile your binary with APP_PLATFORM set to an older API, e.g. android-8. This is a consequence of Android 5.0 enforcing *Position Independent Binaries* (PIE), whereas older devices do not necessarily support PIES. Therefore, you need to use either the PIE or the non-PIE, depending on the device version. If you want to use the binary from within your Android application, you need to check the API level and extract the correct binary.

APP_ABI can be changed to specific platforms such as armeabi to build the binary for those architectures only.

In the worst case, you will have both a PIE and a non-PIE binary for each architecture (about 14 different binaries using ndk-r10e).

To build the executable:

```
cd project  
ndk-build
```

You will find the binaries at project/libs/<architecture>/hello_world. You can use them via ADB (push and chmod it with executable permission) or from your application (extract and chmod it with executable permission).

To determine the architecture of the CPU, retrieve the build property ro.product.cpu.abi for the primary architecture or ro.product.cpu.abi.list (on newer devices) for a complete list of supported architectures. You can do this using the android.os.Build class from within your application or using getprop <name> via ADB.

Section 59.3: How to clean the build

If you need to clean the build:

```
ndk-build clean
```

Section 59.4: How to use a makefile other than Android.mk

```
ndk-build NDK_PROJECT_PATH=PROJECT_PATH APP_BUILD_SCRIPT=MyAndroid.mk
```

第60章：昼夜主题 (AppCompat v23.2 / API 14+)

第60.1节：向应用添加昼夜主题

昼夜主题赋予应用根据一天中的时间和设备的最后已知位置切换配色方案的酷炫功能。

将以下内容添加到你的styles.xml中：

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
    <!-- 在此自定义你的主题。 -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

你可以继承以添加昼夜主题切换功能的主题如下：

- "Theme.AppCompat.DayNight"
- "Theme.AppCompat.DayNight.NoActionBar"
- "Theme.AppCompat.DayNight.DarkActionBar"

除了colorPrimary、colorPrimaryDark和colorAccent之外，你还可以添加任何其他你希望切换的颜色，例如textColorPrimary或textColorSecondary。你也可以将应用的自定义颜色添加到这个样式中。

为了使主题切换生效，你需要在res/values目录下定义一个默认的colors.xml文件，并在res/values-night目录下定义另一个colors.xml文件，分别为白天和夜间定义合适的颜色。

要切换主题，请在你的Java代码中调用AppCompatDelegate.setDefaultNightMode(int)方法。（这将改变整个应用的配色方案，而不仅仅是某个Activity或Fragment。）例如：

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

你可以根据需要传入以下三种之一：

- AppCompatDelegate.MODE_NIGHT_NO：这将设置应用的默认主题，使用res/values目录中定义的颜色。建议该主题使用浅色。
- AppCompatDelegate.MODE_NIGHT_YES：这将为应用设置夜间主题，使用res/values-night目录中定义的颜色。建议该主题使用深色。
- AppCompatDelegate.MODE_NIGHT_AUTO：根据一天中的时间自动切换应用颜色，使用你在values和values-night目录中定义的颜色。

也可以使用getDefaultNightMode()方法获取当前的夜间模式状态。例如：

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

请注意，如果您关闭应用程序后重新打开，主题切换将不会保持。如果您这样做，主题将切换回AppCompatDelegate.MODE_NIGHT_AUTO，这是默认值。如果您希望主题切换保持，请确保将该值存储在共享偏好中，并在应用程序被销毁后每次打开时加载存储的值。

Chapter 60: DayNight Theme (AppCompat v23.2 / API 14+)

Section 60.1: Adding the DayNight theme to an app

The DayNight theme gives an app the cool capability of switching color schemes based on the time of day and the device's last known location.

Add the following to your styles.xml:

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

The themes you can extend from to add day night theme switching capability are the following:

- "Theme.AppCompat.DayNight"
- "Theme.AppCompat.DayNight.NoActionBar"
- "Theme.AppCompat.DayNight.DarkActionBar"

Apart from colorPrimary, colorPrimaryDark and colorAccent, you can also add any other colors that you would like to be switched, e.g. textColorPrimary or textColorSecondary. You can add your app's custom colors to this style as well.

For theme switching to work, you need to define a default colors.xml in the res/values directory and another colors.xml in the res/values-night directory and define day/night colors appropriately.

To switch the theme, call the AppCompatDelegate.setDefaultNightMode(**int**) method from your Java code. (This will change the color scheme for the whole app, not just any one activity or fragment.) For example:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

You can pass any of the following three according to your choice:

- AppCompatDelegate.MODE_NIGHT_NO: this sets the default theme for your app and takes the colors defined in the res/values directory. It is recommended to use light colors for this theme.
- AppCompatDelegate.MODE_NIGHT_YES: this sets a night theme for your app and takes the colors defined in the res/values-night directory. It is recommended to use dark colors for this theme.
- AppCompatDelegate.MODE_NIGHT_AUTO: this auto switches the colors of the app based on the time of the day and the colors you have defined in values and values-night directories.

It is also possible to get the current night mode status using the getDefaultNightMode() method. For example:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Please note, however, that the theme switch will not persist if you kill the app and reopen it. If you do that, the theme will switch back to AppCompatDelegate.MODE_NIGHT_AUTO, which is the default value. If you want the theme switch to persist, make sure you store the value in shared preferences and load the stored value each time the app is opened after it has been destroyed.

第61章：Glide

**** 警告 本文档未维护且经常不准确 ****

Glide的官方文档是更好的资料来源：

关于Glide v4, 请参见<http://bumptech.github.io/glide/>. 关于Glide v3, 请参见<https://github.com/bumptech/glide/wiki>.

第61.1节：加载图片

ImageView

要将指定的URL、Uri、资源ID或任何其他模型加载到ImageView中：

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

对于 Uri, 替换 yourUrl 为你的 Uri (content://media/external/images/1)。对于 Drawable, 替换 yourUrl 为你的资源 ID (R.drawable.image)。

RecyclerView 和 ListView

在 ListView 或 RecyclerView 中, 你可以使用完全相同的代码行：

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

如果你不想在 onBindViewHolder 中启动加载, 确保在修改 ImageView 之前调用 clear() 清除 Glide 可能管理的任何 ImageView :

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // 视图已被清除, 现在你可以安全地设置你自己的资源
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

Chapter 61: Glide

**** WARNING This documentation is unmaintained and frequently inaccurate ****

Glide's official documentation is a much better source:

For Glide v4, see <http://bumptech.github.io/glide/>. For Glide v3, see <https://github.com/bumptech/glide/wiki>.

Section 61.1: Loading an image

ImageView

To load an image from a specified URL, Uri, resource id, or any other model into an ImageView:

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

For Uris, replace yourUrl with your Uri (content://media/external/images/1). For Drawables replace yourUrl with your resource id (R.drawable.image).

RecyclerView and ListView

In ListView or RecyclerView, you can use exactly the same lines:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

If you don't want to start a load in onBindViewHolder, make sure you clear() any ImageView Glide may be managing before modifying the ImageView:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // Now that the view has been cleared, you can safely set your own resource
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

第61.2节：将Glide添加到你的项目中

摘自官方文档：

使用Gradle：

```
repositories {  
    mavenCentral() // jcenter() 也可用，因为它从Maven Central拉取  
}  
  
dependencies {  
    compile 'com.github.bumptech.glide:glide:4.0.0'  
    compile 'com.android.support:support-v4:25.3.1'  
    annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'  
}
```

使用Maven：

```
<dependency>  
    <groupId>com.github.bumptech.glide</groupId>  
    <artifactId>glide</artifactId>  
    <version>4.0.0</version>  
</dependency>  
<dependency>  
    <groupId>com.google.android</groupId>  
    <artifactId>support-v4</artifactId>  
    <version>r7</version>  
</dependency>  
<dependency>  
    <groupId>com.github.bumptech.glide</groupId>  
    <artifactId>compiler</artifactId>  
    <version>4.0.0</version>  
    <optional>true</optional>  
</dependency>
```

根据您的 ProGuard (DexGuard) 配置和使用情况，您可能还需要在您的 proguard.cfg 中包含以下行（更多信息请参见 [Glide 的 wiki](#)）：

```
-keep public class * implements com.bumptech.glide.module.GlideModule  
-keep public class * extends com.bumptech.glide.AppGlideModule  
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$** {  
    **[] $VALUES;  
    public *;  
}  
  
# 仅限 DexGuard 使用  
-keepresourceelements manifest/application/meta-data@value=GlideModule
```

第 61.3 节：Glide 圆形变换（在圆形 ImageView 中加载图片）

使用 Glide 创建圆形图片。

```
public class CircleTransform extends BitmapTransformation {  
  
    public CircleTransform(Context context) {  
        super(context);  
    }
```

Section 61.2: Add Glide to your project

From the [official documentation](#):

With Gradle:

```
repositories {  
    mavenCentral() // jcenter() works as well because it pulls from Maven Central  
}  
  
dependencies {  
    compile 'com.github.bumptech.glide:glide:4.0.0'  
    compile 'com.android.support:support-v4:25.3.1'  
    annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'  
}
```

With Maven:

```
<dependency>  
    <groupId>com.github.bumptech.glide</groupId>  
    <artifactId>glide</artifactId>  
    <version>4.0.0</version>  
</dependency>  
<dependency>  
    <groupId>com.google.android</groupId>  
    <artifactId>support-v4</artifactId>  
    <version>r7</version>  
</dependency>  
<dependency>  
    <groupId>com.github.bumptech.glide</groupId>  
    <artifactId>compiler</artifactId>  
    <version>4.0.0</version>  
    <optional>true</optional>  
</dependency>
```

Depending on your ProGuard (DexGuard) config and usage, you may also need to include the following lines in your proguard.cfg (See [Glide's wiki](#) for more info):

```
-keep public class * implements com.bumptech.glide.module.GlideModule  
-keep public class * extends com.bumptech.glide.AppGlideModule  
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$** {  
    **[] $VALUES;  
    public *;  
}  
  
# for DexGuard only  
-keepresourceelements manifest/application/meta-data@value=GlideModule
```

Section 61.3: Glide circle transformation (Load image in a circular ImageView)

Create a circle image with glide.

```
public class CircleTransform extends BitmapTransformation {  
  
    public CircleTransform(Context context) {  
        super(context);  
    }
```

```

@Override protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int outWidth, int
outHeight) {
    return circleCrop(pool, toTransform);
}

private static Bitmap circleCrop(BitmapPool pool, Bitmap source) {
    if (source == null) return null;

    int size = Math.min(source.getWidth(), source.getHeight());
    int x = (source.getWidth() - size) / 2;
    int y = (source.getHeight() - size) / 2;

    Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

    Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
    if (result == null) {
        result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
    }

    Canvas canvas = new Canvas(result);
    Paint paint = new Paint();
    paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
        BitmapShader.TileMode.CLAMP));
    paint.setAntiAlias(true);
    float r = size / 2f;
    canvas.drawCircle(r, r, r, paint);
    return result;
}

@Override public String getId() {
    return getClass().getName();
}
}

```

用法：

```

Glide.with(context)
.load(yourimageurl)
.transform(new CircleTransform(context))
.into(userImageView);

```

第61.4节：默认变换

Glide包含两种默认变换，fit center（适应中心）和center crop（中心裁剪）。

适应中心：

```

Glide.with(context)
.load(yourUrl)
.fitCenter()
.into(yourView);

```

适应中心执行的变换与Android的ScaleType.FIT_CENTER相同。

中心裁剪：

```

Glide.with(context)
.load(yourUrl)
.centerCrop();

```

```

@Override protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int outWidth, int
outHeight) {
    return circleCrop(pool, toTransform);
}

private static Bitmap circleCrop(BitmapPool pool, Bitmap source) {
    if (source == null) return null;

    int size = Math.min(source.getWidth(), source.getHeight());
    int x = (source.getWidth() - size) / 2;
    int y = (source.getHeight() - size) / 2;

    Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

    Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
    if (result == null) {
        result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
    }

    Canvas canvas = new Canvas(result);
    Paint paint = new Paint();
    paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
        BitmapShader.TileMode.CLAMP));
    paint.setAntiAlias(true);
    float r = size / 2f;
    canvas.drawCircle(r, r, r, paint);
    return result;
}

@Override public String getId() {
    return getClass().getName();
}
}

```

Usage:

```

Glide.with(context)
.load(yourimageurl)
.transform(new CircleTransform(context))
.into(userImageView);

```

Section 61.4: Default transformations

Glide includes two default transformations, fit center and center crop.

Fit center:

```

Glide.with(context)
.load(yourUrl)
.fitCenter()
.into(yourView);

```

Fit center performs the same transformation as Android's [ScaleType.FIT_CENTER](#).

Center crop:

```

Glide.with(context)
.load(yourUrl)
.centerCrop();

```

```
.进入(yourView);
```

中心裁剪执行的变换与Android的ScaleType.CENTER_CROP相同。

更多信息，请参见Glide的[wiki](#)。

第61.5节：使用自定义Glide目标实现Glide圆角图片

首先创建工具类或在需要的类中使用此方法

```
public class UIUtils {  
    public static BitmapImageViewTarget 获取圆角图片目标(@NonNull final Context context, @NonNull  
    final ImageView imageView,  
                           final float radius) {  
        return new BitmapImageViewTarget(imageView) {  
            @Override  
            protected void setResource(@NonNull Bitmap resource) {  
                RoundedBitmapDrawable circularBitmapDrawable =  
                    RoundedBitmapDrawableFactory.create(context.getResources(), resource);  
                circularBitmapDrawable.setCornerRadius(radius);  
                imageView.setImageDrawable(circularBitmapDrawable);  
            }  
        };  
    }  
}
```

加载图片：

```
Glide.with(context)  
.load(imageUrl)  
.asBitmap()  
.into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

因为你使用了 `asBitmap()`，动画将被移除。不过你可以在这里使用 `animate()` 方法来实现自定义动画。

示例：与默认 Glide 动画类似的淡入效果。

```
Glide.with(context)  
.load(imageUrl)  
.asBitmap()  
.animate(R.anim.abc_fade_in)  
.into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

请注意，此动画是支持库的私有资源——不建议使用，因为它可能会更改甚至被移除。

请注意，使用RoundedBitmapDrawableFactory还需要支持库

第61.6节：占位符和错误处理

如果你想在加载过程中显示一个Drawable，可以添加一个占位符：

```
Glide.with(context)  
.load(yourUrl)  
.placeholder(R.drawable.placeholder)
```

```
.into(yourView);
```

Center crop performs the same transformation as Android's [ScaleType.CENTER_CROP](#).

For more information, see [Glide's wiki](#).

Section 61.5: Glide rounded corners image with custom Glide target

First make utility class or use this method in class needed

```
public class UIUtils {  
    public static BitmapImageViewTarget getRoundedImageTarget(@NonNull final Context context, @NonNull  
    final ImageView imageView,  
                           final float radius) {  
        return new BitmapImageViewTarget(imageView) {  
            @Override  
            protected void setResource(@NonNull Bitmap resource) {  
                RoundedBitmapDrawable circularBitmapDrawable =  
                    RoundedBitmapDrawableFactory.create(context.getResources(), resource);  
                circularBitmapDrawable.setCornerRadius(radius);  
                imageView.setImageDrawable(circularBitmapDrawable);  
            }  
        };  
    }  
}
```

Loading image:

```
Glide.with(context)  
.load(imageUrl)  
.asBitmap()  
.into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Because you use `asBitmap()` the animations will be removed though. You can use your own animation in this place using the `animate()` method.

Example with similar fade in to default Glide animation.

```
Glide.with(context)  
.load(imageUrl)  
.asBitmap()  
.animate(R.anim.abc_fade_in)  
.into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Please note this animation is support library private resource - it is unrecommended to use as it can change or even be removed.

Note you also need to have support library to use [RoundedBitmapDrawableFactory](#)

Section 61.6: Placeholder and Error handling

If you want to add a Drawable be shown during the load, you can add a placeholder:

```
Glide.with(context)  
.load(yourUrl)  
.placeholder(R.drawable.placeholder)
```

```
.into(imageView);
```

如果你想在加载失败时显示一个Drawable：

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

如果你想在提供空模型（URL、Uri、文件路径等）时显示一个Drawable：

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

第61.7节：预加载图片

预加载远程图片并确保图片只下载一次：

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

然后：

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // 这里也适用全部
    .into(imageView);
```

预加载本地图片并确保转换后的副本存在于磁盘缓存中（也可能存在于内存缓存中）：

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // 或者你想要的任何转换
    .preload(200, 200); // 或者你想要的宽度和高度
```

然后：

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // 你必须使用与上面相同的转换
    .override(200, 200) // 你必须使用与上面相同的宽度和高度
    .into(imageView);
```

第61.8节：处理Glide图片加载失败

```
Glide
    .with(context)
    .load(currentUrl)
    .into(new BitmapImageViewTarget(profilePicture) {
        @Override
        protected void setResource(Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapFactory.create(context.getResources(), resource);
```

```
.into(imageView);
```

If you want a Drawable to be shown if the load fails for any reason:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

If you want a Drawable to be shown if you provide a null model (URL, Uri, file path etc):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Section 61.7: Preloading images

To preload remote images and ensure that the image is only downloaded once:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Then:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

To preload local images and make sure a transformed copy is in the disk cache (and maybe the memory cache):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Then:

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // You must use the same transformation as above
    .override(200, 200) // You must use the same width and height as above
    .into(imageView);
```

Section 61.8: Handling Glide image load failed

```
Glide
    .with(context)
    .load(currentUrl)
    .into(new BitmapImageViewTarget(profilePicture) {
        @Override
        protected void setResource(Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapFactory.create(context.getResources(), resource);
```

```

circularBitmapDrawable.setCornerRadius(radius);
    imageView.setImageDrawable(circularBitmapDrawable);
}

@Override
public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
    super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
Log.e(TAG, e.getMessage(), e);
}
);

```

在 SET_YOUR_DEFAULT_IMAGE 位置，您可以设置任何默认的 Drawable。如果图片加载失败，将显示此图片。

第61.9节：在圆形ImageView中加载图像，无需自定义变换

创建一个自定义的 BitmapImageViewTarget 来加载图片：

```

public class CircularBitmapImageViewTarget extends BitmapImageViewTarget
{
    private Context context;
    private ImageView imageView;

    public CircularBitmapImageViewTarget(Context context, ImageView imageView)
    {
        super(imageView);
        this.context = context;
        this.imageView = imageView;
    }

    @Override
    protected void setResource(Bitmap resource)
    {
        RoundedBitmapDrawable bitmapDrawable =
RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        bitmapDrawable.setCircular(true);
        imageView.setImageDrawable(bitmapDrawable);
    }
}

```

用法：

```

Glide
.with(context)
    .load(yourimageidentifier)
    .asBitmap()
.into(new CircularBitmapImageViewTarget(context, imageView));

```

```

circularBitmapDrawable.setCornerRadius(radius);
    imageView.setImageDrawable(circularBitmapDrawable);
}

@Override
public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
    super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
Log.e(TAG, e.getMessage(), e);
}
);

```

Here at SET_YOUR_DEFAULT_IMAGE place you can set any default Drawable. This image will be shown if Image loading is failed.

Section 61.9: Load image in a circular ImageView without custom transformations

Create a custom BitmapImageViewTarget to load the image into:

```

public class CircularBitmapImageViewTarget extends BitmapImageViewTarget
{
    private Context context;
    private ImageView imageView;

    public CircularBitmapImageViewTarget(Context context, ImageView imageView)
    {
        super(imageView);
        this.context = context;
        this.imageView = imageView;
    }

    @Override
    protected void setResource(Bitmap resource)
    {
        RoundedBitmapDrawable bitmapDrawable =
RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        bitmapDrawable.setCircular(true);
        imageView.setImageDrawable(bitmapDrawable);
    }
}

```

Usage:

```

Glide
.with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));

```

第62章：对话框

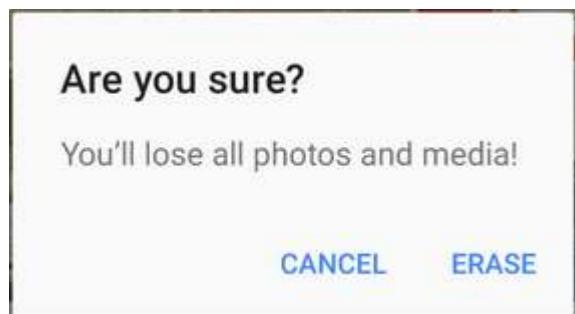
行	描述
show();	显示对话框
setContentView(R.layout.yourlayout);	将对话框的内容视图设置为你的自定义布局。
dismiss()	关闭对话框

第62.1节：使用Appcompat向你的应用添加Material Design风格的AlertDialog

AlertDialog 是 Dialog 的子类，可以显示一个、两个或三个按钮。如果您只想在此对话框中显示一个字符串，请使用 setMessage() 方法。

来自android.app包的AlertDialog在不同的Android操作系统版本上显示效果不同。

Android V7 Appcompat库提供了一个AlertDialog实现，可以在所有支持的Android操作系统版本上以Material Design样式显示，如下所示：



首先你需要将V7 Appcompat库添加到你的项目中。你可以在应用级别的build.gradle文件中这样做：

```
dependencies {  
    compile 'com.android.support:appcompat-v7:24.2.1'  
    //.....  
}
```

确保导入正确的类：

```
import android.support.v7.app.AlertDialog;
```

然后像这样创建AlertDialog：

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle("你确定吗？");  
builder.setMessage("你将丢失所有照片和媒体！");  
builder.setPositiveButton("删除", null);  
builder.setNegativeButton("取消", null);  
builder.show();
```

第62.2节：一个基本的警告对话框

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);  
//设置标题  
builder.setTitle("重置...")  
//设置消息  
.setMessage("你确定吗？")
```

Chapter 62: Dialog

Line	Description
show();	Shows the dialog
setContentView(R.layout.yourlayout);	sets the ContentView of the dialog to your custom layout.
dismiss()	Closes the dialog

Section 62.1: Adding Material Design AlertDialog to your app using Appcompat

AlertDialog is a subclass of Dialog that can display one, two or three buttons. If you only want to display a String in this dialog box, use the setMessage() method.

The AlertDialog from android.app package displays differently on different Android OS Versions.

The Android V7 Appcompat library provides an AlertDialog implementation which will display with Material Design on all supported Android OS versions, as shown below:



First you need to add the V7 Appcompat library to your project. you can do this in the app level build.gradle file:

```
dependencies {  
    compile 'com.android.support:appcompat-v7:24.2.1'  
    //.....  
}
```

Be sure to import the correct class:

```
import android.support.v7.app.AlertDialog;
```

Then Create AlertDialog like this:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle("Are you sure?");  
builder.setMessage("You'll lose all photos and media!");  
builder.setPositiveButton("ERASE", null);  
builder.setNegativeButton("CANCEL", null);  
builder.show();
```

Section 62.2: A Basic Alert Dialog

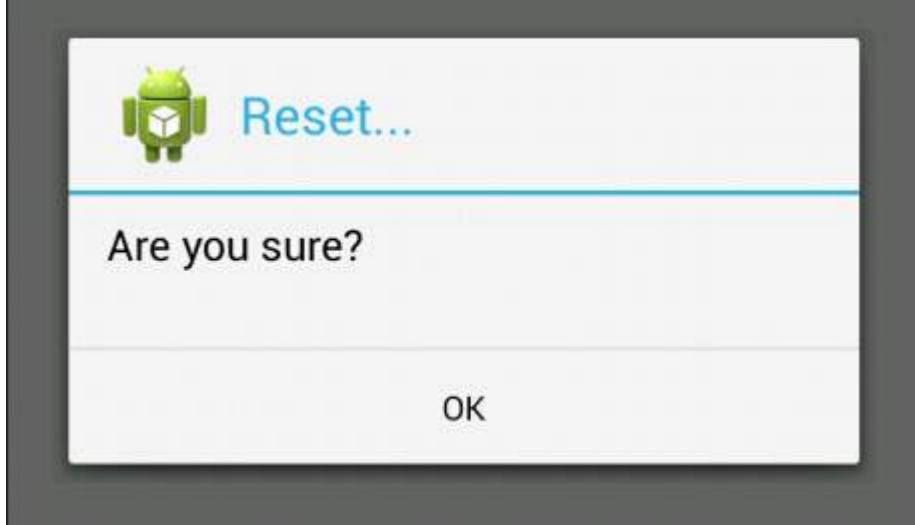
```
AlertDialog.Builder builder = new AlertDialog.Builder(context);  
//Set Title  
builder.setTitle("Reset...")  
//Set Message  
.setMessage("Are you sure?")
```

```

//设置对话框图标
.setIcon(drawable)
    //设置正面按钮，在本例中为“确定”，点击后将关闭对话框并执行onClick方法中的所有操作
.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        // 重置
    }
});
AlertDialog dialog = builder.create();
    // 现在，您可以随时调用：
dialog.show();
    // 这样您就可以显示对话框。

```

现在这段代码将实现以下功能：



(图片来源：WikiHow)

第62.3节：AlertDialog中的ListView

我们通常可以使用ListView或RecyclerView来从项目列表中选择，但如果选项较少且希望用户从中选择一个，我们可以使用AlertDialog.Builder的setAdapter方法。

```

private void showDialog()
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("选择任意项目");

    final List<String> 标签 = new ArrayList<>();
    标签.add("项目 1");
    标签.add("项目 2");
    标签.add("项目 3");
    标签.add("项目 4");

    ArrayAdapter<String> 数据适配器 = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, 标签);
    builder.setAdapter(数据适配器, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this, "你选择了 " +
                标签.get(which), Toast.LENGTH_LONG).show();
        }
    });
    AlertDialog dialog = builder.create();
}

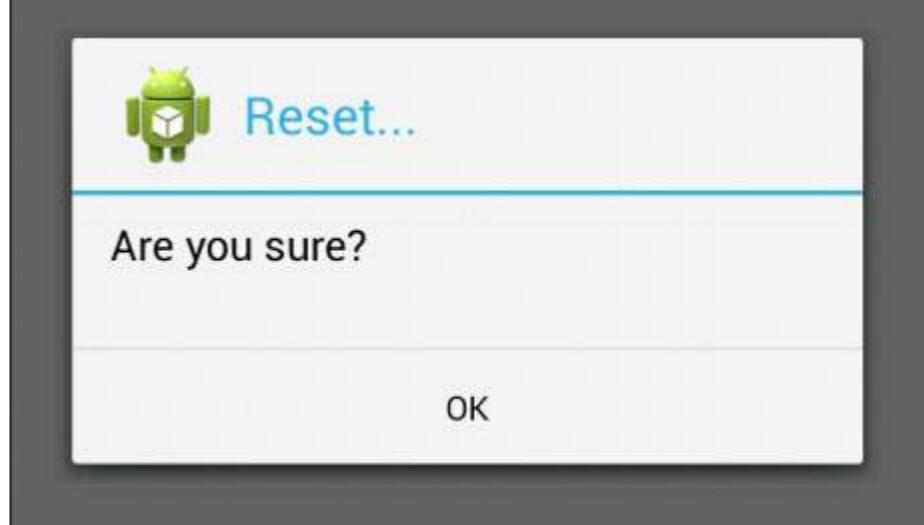
```

```

//Set the icon of the dialog
.setIcon(drawable)
    //Set the positive button, in this case, OK, which will dismiss the dialog and do
everything in the onClick method
.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        // Reset
    }
});
AlertDialog dialog = builder.create();
//Now, any time you can call on:
dialog.show();
//So you can show the dialog.

```

Now this code will achieve this:



(Image source: WikiHow)

Section 62.3: ListView in AlertDialog

We can always use [ListView](#) or [RecyclerView](#) for selection from list of items, but if we have small amount of choices and among those choices we want user to select one, we can use [AlertDialog.Builder](#) `setAdapter`.

```

private void showDialog()
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Choose any item");

    final List<String> lables = new ArrayList<>();
    lables.add("Item 1");
    lables.add("Item 2");
    lables.add("Item 3");
    lables.add("Item 4");

    ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, lables);
    builder.setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this, "You have selected " +
                lables.get(which), Toast.LENGTH_LONG).show();
        }
    });
    AlertDialog dialog = builder.create();
}

```

```
dialog.show();
}
```

也许，如果我们不需要任何特定的ListView，我们可以使用一种基本的方法：

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("选择一个项目")
.setItems(R.array.your_array, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // 'which' 参数包含所选项目的索引位置
        Log.v(TAG, "选中位置的项目 " + which);
    }
});
builder.create().show();
```

```
dialog.show();
}
```

Perhaps, if we don't need any particular ListView, we can use a basic way:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Select an item")
.setItems(R.array.your_array, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // The 'which' argument contains the index position of the selected item
        Log.v(TAG, "Selected item on position " + which);
    }
});
builder.create().show();
```

第62.4节：带有EditText的自定义警告对话框

```
void alertDialogDemo() {
    // 获取 alert_dialog.xml 视图
    LayoutInflator li = LayoutInflator.from(getApplicationContext());
    View promptsView = li.inflate(R.layout.alert_dialog, null);

    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
        getApplicationContext());

    // 将 alert_dialog.xml 设置给 alertDialog 构建器
    alertDialogBuilder.setView(promptsView);

    final EditText userInput = (EditText) promptsView.findViewById(R.id.etUserInput);

    // 设置对话框消息
    alertDialogBuilder
.setCancelable(false)
.setPositiveButton("确定", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        // 获取用户输入并设置到结果中
        // 编辑文本框
        Toast.makeText(getApplicationContext(), "Entered:
" + userInput.getText().toString(), Toast.LENGTH_LONG).show();
    }
})
.setNegativeButton("取消",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            dialog.cancel();
        }
});

    // 创建警告对话框
    AlertDialog alertDialog = alertDialogBuilder.create();

    // 显示它
    alertDialog.show();
}
```

Xml 文件 : res/layout/alert_dialog.xml

```
<TextView
    android:id="@+id/textView1"
```

Section 62.4: Custom Alert Dialog with EditText

```
void alertDialogDemo() {
    // get alert_dialog.xml view
    LayoutInflator li = LayoutInflator.from(getApplicationContext());
    View promptsView = li.inflate(R.layout.alert_dialog, null);

    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
        getApplicationContext());

    // set alert_dialog.xml to alertDialog builder
    alertDialogBuilder.setView(promptsView);

    final EditText userInput = (EditText) promptsView.findViewById(R.id.etUserInput);

    // set dialog message
    alertDialogBuilder
.setCancelable(false)
.setPositiveButton("OK", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        // get user input and set it to result
        // edit text
        Toast.makeText(getApplicationContext(), "Entered:
" + userInput.getText().toString(), Toast.LENGTH_LONG).show();
    }
})
.setNegativeButton("Cancel",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            dialog.cancel();
        }
});

    // create alert dialog
    AlertDialog alertDialog = alertDialogBuilder.create();

    // show it
    alertDialog.show();
}
```

Xml file: res/layout/alert_dialog.xml

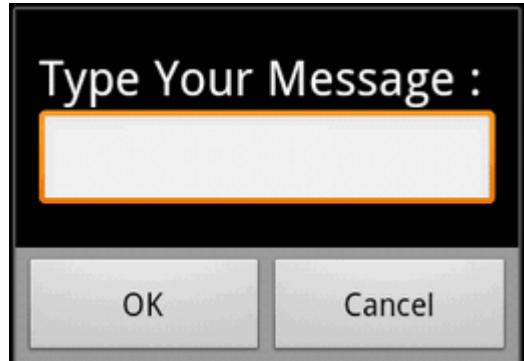
```
<TextView
    android:id="@+id/textView1"
```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="请输入您的信息 :"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />
</EditText>

```



第62.5节：日期选择对话框

DatePickerDialog 是使用 DatePicker 的最简单方式，因为可以在应用的任何地方显示对话框。你不必自己实现带有 DatePicker 控件的布局。

如何显示对话框：

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

您可以从上方的对话框中获取DatePicker控件，以便访问更多功能，例如设置以毫秒为单位的最小日期：

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

第62.6节：DatePicker

DatePicker 允许用户选择日期。当我们创建新的DatePicker实例时，可以设置初始日期。如果不设置初始日期，默认会设置为当前日期。

我们可以通过使用DatePickerDialog或者创建带有DatePicker

我们还可以限制用户可选择的日期范围。

通过设置以毫秒为单位的最小日期

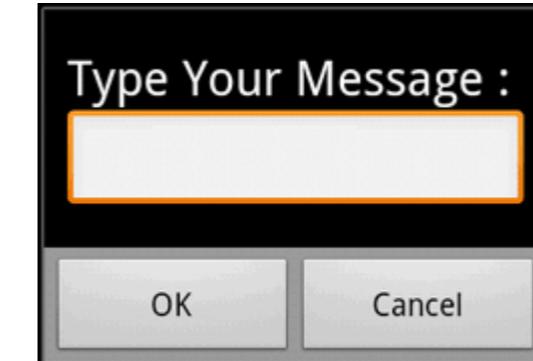
```
//在这种情况下，用户只能选择未来的日期
```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Type Your Message :"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />
</EditText>

```



Section 62.5: DatePickerDialog

DatePickerDialog is the simplest way to use DatePicker, because you can show dialog anywhere in your app. You don't have to implement your own layout with DatePicker widget.

How to show dialog:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

You can get DatePicker widget from dialog above, to get access to more functions, and for example set minimum date in milliseconds:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Section 62.6: DatePicker

DatePicker allows user to pick date. When we create new instance of DatePicker, we can set initial date. If we don't set initial date, current date will be set by default.

We can show DatePicker to user by using DatePickerDialog or by creating our own layout with DatePicker widget.

Also we can limit range of date, which user can pick.

By setting minimum date in milliseconds

```
//In this case user can pick date only from future
```

```
datePicker.setMinDate(System.currentTimeMillis());
```

通过设置以毫秒为单位的最大日期

```
//在这种情况下，用户只能选择日期，且只能选择当前日期之后一周内的日期。  
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

要接收用户选择的日期信息，我们必须使用Listener。

如果我们使用DatePickerDialog，可以在创建新的

DatePickerDialog实例时，在构造函数中设置OnDateSetListener：

示例使用DatePickerDialog

```
public class SampleActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        ...  
    }  
  
    private void showDatePicker() {  
        //我们需要使用日历来设置DatePickerDialog中的当前日期作为初始日期。  
        Calendar calendar = new GregorianCalendar(Locale.getDefault());  
        int year = calendar.get(Calendar.YEAR);  
        int month = calendar.get(Calendar.MONTH);  
        int day = calendar.get(Calendar.DAY_OF_MONTH);  
  
        DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year, month, day);  
        datePickerDialog.show();  
    }  
  
    @Override  
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {  
    }  
}
```

否则，如果我们使用DatePicker控件创建自己的布局，也必须创建自己的监听器，正如其他示例中所示

第62.7节：警告对话框

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(  
    MainActivity.this);  
  
alertDialogBuilder.setTitle("标题对话框");  
    alertDialogBuilder  
.setMessage("消息对话框")  
.setCancelable(true)  
.setPositiveButton("是",  
    new DialogInterface.OnClickListener() {  
  
        public void onClick(DialogInterface dialog, int arg1) {  
            // 处理确定按钮  
    }}
```

```
datePicker.setMinDate(System.currentTimeMillis());
```

By setting maximum date in milliseconds

```
//In this case user can pick date only, before following week.  
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

To receive information, about which date was picked by user, we have to use Listener.

If we are using DatePickerDialog, we can set OnDateSetListener in constructor when we are creating new instance of DatePickerDialog:

Sample use of DatePickerDialog

```
public class SampleActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        ...  
    }  
  
    private void showDatePicker() {  
        //We need calendar to set current date as initial date in DatePickerDialog.  
        Calendar calendar = new GregorianCalendar(Locale.getDefault());  
        int year = calendar.get(Calendar.YEAR);  
        int month = calendar.get(Calendar.MONTH);  
        int day = calendar.get(Calendar.DAY_OF_MONTH);  
  
        DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year, month, day);  
        datePickerDialog.show();  
    }  
  
    @Override  
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {  
    }  
}
```

Otherwise, if we are creating our own layout with DatePicker widget, we also have to create our own listener as it was shown in other example

Section 62.7: Alert Dialog

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(  
    MainActivity.this);  
  
alertDialogBuilder.setTitle("Title Dialog");  
    alertDialogBuilder  
.setMessage("Message Dialog")  
.setCancelable(true)  
.setPositiveButton("Yes",  
    new DialogInterface.OnClickListener() {  
  
        public void onClick(DialogInterface dialog, int arg1) {  
            // Handle Positive Button  
    }}
```

```

        }
    .setNegativeButton("否",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int arg1) {
                // 处理否定按钮
                dialog.cancel();
            }
        });
}

AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();

```

第62.8节：带多行标题的警告对话框

AlertDialog.Builder的setCustomTitle()方法允许您指定一个任意视图作为对话框标题。此方法的一个常见用途是构建具有较长标题的警告对话框。

```

AlertDialog.Builder builder = new AlertDialog.Builder(context, Theme_Material_Light_Dialog);
builder.setCustomTitle(LayoutInflater.inflate(context, R.layout.my_dialog_title, null))
.setView(LayoutInflater.inflate(context, R.layout.my_dialog, null))
.setPositiveButton("确定", null);

Dialog dialog = builder.create();
dialog.show();

```

my_dialog_title.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="罗伦·伊普苏姆 (Lorem ipsum) 是虚拟文本，用于排版和设计。Curabitur
tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor
iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla
tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo
pharetra semper faucibus vel velit."
        android:textStyle="bold"/>

</LinearLayout>

```

my_dialog.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

```

```

        }
    .setNegativeButton("No",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int arg1) {
                // Handle Negative Button
                dialog.cancel();
            }
        });
}

AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();

```

Section 62.8: Alert Dialog with Multi-line Title

The setCustomTitle() method of AlertDialog.Builder lets you specify an arbitrary view to be used for the dialog title. One common use for this method is to build an alert dialog that has a long title.

```

AlertDialog.Builder builder = new AlertDialog.Builder(context, Theme_Material_Light_Dialog);
builder.setCustomTitle(LayoutInflater.inflate(context, R.layout.my_dialog_title, null))
.setView(LayoutInflater.inflate(context, R.layout.my_dialog, null))
.setPositiveButton("OK", null);

Dialog dialog = builder.create();
dialog.show();

```

my_dialog_title.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

```

```

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur
tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor
iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla
tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo
pharetra semper faucibus vel velit."
        android:textStyle="bold"/>

</LinearLayout>

```

my_dialog.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

```

```
    android:orientation="vertical"
    android:padding="16dp"
    android:scrollbars="vertical"

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

</LinearLayout>
</ScrollView>
```

```
    android:orientation="vertical"
    android:padding="16dp"
    android:scrollbars="vertical"

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Hello world again!"/>

</LinearLayout>
</ScrollView>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo pharetra semper faucibus vel velit.

Hello world!

Hello world again!

Hello world again!

Hello world again!

OK

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo pharetra semper faucibus vel velit.

Hello world!

Hello world again!

Hello world again!

Hello world again!

OK

第62.9节：DialogFragment中的日期选择器

对话框的xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ACCEPT"
        android:id="@+id/buttonAccept" />

</LinearLayout>
```

对话框类：

```
public class ChooseDate extends DialogFragment implements View.OnClickListener {

    private DatePicker datePicker;
    private Button acceptButton;

    private boolean isDateSetted = false;
    private int year;
```

Section 62.9: Date Picker within DialogFragment

xml of the Dialog:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ACCEPT"
        android:id="@+id/buttonAccept" />

</LinearLayout>
```

Dialog Class:

```
public class ChooseDate extends DialogFragment implements View.OnClickListener {

    private DatePicker datePicker;
    private Button acceptButton;

    private boolean isDateSetted = false;
    private int year;
```

```

private int month;
private int day;

private DateListener listener;

public interface DateListener {
    onDateSelected(int year, int month, int day);
}

public ChooseDate(){}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.dialog_year_picker, container);

    getDialog().setTitle(getResources().getString("TITLE"));

    datePicker = (DatePicker) rootView.findViewById(R.id.datePicker);
    acceptButton = (Button) rootView.findViewById(R.id.buttonAccept);
    acceptButton.setOnClickListener(this);

    if (isDateSetted) {
        datePicker.updateDate(year, month, day);
    }

    return rootView;
}

@Override
public void onClick(View v) {
    switch(v.getId()){
        case R.id.buttonAccept:
            int year = datePicker.getYear();
            int month = datePicker.getMonth() + 1; // 月份从0开始
            int day = datePicker.getDayOfMonth();

            listener.onDateSelected(year, month, day);
            break;
    }
    this.dismiss();
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    listener = (DateListener) context;
}

public void setDate(int year, int month, int day) {

    this.year = year;
    this.month = month;
    this.day = day;
    this.isDateSetted = true;
}
}

```

调用对话框的活动：

```

private int month;
private int day;

private DateListener listener;

public interface DateListener {
    onDateSelected(int year, int month, int day);
}

public ChooseDate(){}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.dialog_year_picker, container);

    getDialog().setTitle(getResources().getString("TITLE"));

    datePicker = (DatePicker) rootView.findViewById(R.id.datePicker);
    acceptButton = (Button) rootView.findViewById(R.id.buttonAccept);
    acceptButton.setOnClickListener(this);

    if (isDateSetted) {
        datePicker.updateDate(year, month, day);
    }

    return rootView;
}

@Override
public void onClick(View v) {
    switch(v.getId()){
        case R.id.buttonAccept:
            int year = datePicker.getYear();
            int month = datePicker.getMonth() + 1; // months start in 0
            int day = datePicker.getDayOfMonth();

            listener.onDateSelected(year, month, day);
            break;
    }
    this.dismiss();
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    listener = (DateListener) context;
}

public void setDate(int year, int month, int day) {

    this.year = year;
    this.month = month;
    this.day = day;
    this.isDateSetted = true;
}
}

```

Activity calling the dialog:

```

public class MainActivity extends AppCompatActivity implements ChooseDate.DateListener{

    private int year;
    private int month;
    private int day;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        private void showAlertDialog();
    }

    private void showAlertDialog(){
        ChooseDate pickDialog = new ChooseDate();
        // 我们可以设置一个日期
        // pickDialog.setDate(23, 10, 2016);
        pickDialog.show(getFragmentManager(), "");
    }

    @Override
    onDateSelected(int year, int month, int day){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}

```

```

public class MainActivity extends AppCompatActivity implements ChooseDate.DateListener{

    private int year;
    private int month;
    private int day;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        private void showAlertDialog();
    }

    private void showAlertDialog(){
        ChooseDate pickDialog = new ChooseDate();
        // We could set a date
        // pickDialog.setDate(23, 10, 2016);
        pickDialog.show(getFragmentManager(), "");
    }

    @Override
    onDateSelected(int year, int month, int day){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}

```

第62.10节：无背景无标题的全屏自定义对话框

在styles.xml中添加你的自定义样式：

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
        </style>
    </resources>

```

为对话框创建自定义布局：fullscreen.xml：

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>

```

然后在java文件中你可以将其用于Activity或Dialog等：

```

import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

Section 62.10: Fullscreen Custom Dialog with no background and no title

in styles.xml add your custom style:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
        </style>
    </resources>

```

Create your custom layout for the dialog: fullscreen.xml:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>

```

Then in java file you can use it for an Activity or Dialog etc:

```

import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```
//您可以为该活动设置无内容。  
Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);  
mDialog.setContentView(R.layout.fullscreen);  
mDialog.show();  
}  
}
```

```
//You can set no content for the activity.  
Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);  
mDialog.setContentView(R.layout.fullscreen);  
mDialog.show();  
}  
}
```

第63章：增强警告对话框

本主题关于为AlertDialog添加额外功能进行增强。

第63.1节：包含可点击链接的警告对话框

为了显示一个包含可点击打开链接的警告对话框，您可以使用以下代码：

```
AlertDialog.Builder builder1 = new AlertDialog.Builder(youractivity.this);

builder1.setMessage(Html.fromHtml("your message,<a href=\"http://www.google.com\">link</a>"));

builder1.setCancelable(false);
builder1.setPositiveButton("ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
    }
});

AlertDialog Alert1 = builder1.create();
Alert1.show();
((TextView)Alert1.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance());
```

Chapter 63: Enhancing Alert Dialogs

This topic is about enhancing an [AlertDialog](#) with additional features.

Section 63.1: Alert dialog containing a clickable link

In order to show an alert dialog containing a link which can be opened by clicking it, you can use the following code:

```
AlertDialog.Builder builder1 = new AlertDialog.Builder(youractivity.this);

builder1.setMessage(Html.fromHtml("your message,<a href=\"http://www.google.com\">link</a>"));

builder1.setCancelable(false);
builder1.setPositiveButton("ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
    }
});

AlertDialog Alert1 = builder1.create();
Alert1.show();
((TextView)Alert1.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance());
```

第64章：动画AlertDialog框

带有动画效果的动画AlertDialog。您可以为对话框添加一些动画效果，如淡入、左滑、上滑、下滑、右滑、下落、报纸翻页、水平翻转、垂直翻转、底部旋转、左旋转、切割、摇晃、侧填充，使您的应用更具吸引力。

第64.1节：为动画对话框添加以下代码。

animated_android_dialog_box.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1184be"
        android:onClick="animatedDialog1"
        android:text="动画下落对话框"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog2"
        android:text="动画材质翻转对话框"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1184be"
        android:onClick="animatedDialog3"
        android:text="动画材质抖动对话框"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog4"
        android:text="动画顶部滑动对话框"
        android:textColor="#fff" />
```

AnimatedAndroidDialogExample.java

```
public class AnimatedAndroidDialogExample extends AppCompatActivity {

    NiftyDialogBuilder materialDesignAnimatedDialog;

    @Override
```

Chapter 64: Animated AlertDialog Box

Animated Alert Dialog Which display with some animation effects.. You Can Get Some Animation for dialog box like Fadein, Slideleft, Slidetop, SlideBottom, Slideright, Fall, Newspaper, Flip, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill to make Your application attractive..

Section 64.1: Put Below code for Animated dialog..

animated_android_dialog_box.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1184be"
        android:onClick="animatedDialog1"
        android:text="Animated Fall Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog2"
        android:text="Animated Material Flip Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog3"
        android:text="Animated Material Shake Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog4"
        android:text="Animated Slide Top Dialog"
        android:textColor="#fff" />
```

AnimatedAndroidDialogExample.java

```
public class AnimatedAndroidDialogExample extends AppCompatActivity {

    NiftyDialogBuilder materialDesignAnimatedDialog;

    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.animated_android_dialog_box);

    materialDesignAnimatedDialog = NiftyDialogBuilder.getInstance(this);
}

public void animatedDialog1(View view) {
    materialDesignAnimatedDialog
    .withTitle("动画下落对话框标题")
        .withMessage("在此添加您的对话框消息。动画对话框描述位置。")
        .withDialogColor("#FFFFFF")
    .withButton1Text("确定")
        .withButton2Text("取消")
    .withDuration(700)
    .withEffect(Effectstype.Fall)
        .show();
}

public void animatedDialog2(View view) {
    materialDesignAnimatedDialog
    .withTitle("动画翻转对话框标题")
        .withMessage("在此添加您的对话框消息。动画对话框描述位置。")
        .withDialogColor("#1c90ec")
    .withButton1Text("确定")
        .withButton2Text("取消")
    .withDuration(700)
    .withEffect(Effectstype.Flip)
        .show();
}

public void animatedDialog3(View view) {
    materialDesignAnimatedDialog
    .withTitle("动画抖动对话框标题")
        .withMessage("在此添加您的对话框消息。动画对话框描述位置。")
        .withDialogColor("#1c90ec")
    .withButton1Text("确定")
        .withButton2Text("取消")
    .withDuration(700)
    .withEffect(Effectstype.Shake)
        .show();
}

public void animatedDialog4(View view) {
    materialDesignAnimatedDialog
    .withTitle("动画从顶部滑动对话框标题")
        .withMessage("在此添加您的对话框消息。动画对话框描述位置。")
        .withDialogColor("#1c90ec")
    .withButton1Text("确定")
        .withButton2Text("取消")
    .withDuration(700)
    .withEffect(Effectstype.Slidetop)
        .show();
}

```

在您的 build.gradle 中添加以下行以包含 NifyBuilder(CustomView)

build.gradle

```
dependencies {
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.animated_android_dialog_box);

    materialDesignAnimatedDialog = NiftyDialogBuilder.getInstance(this);
}

public void animatedDialog1(View view) {
    materialDesignAnimatedDialog
    .withTitle("Animated Fall Dialog Title")
        .withMessage("Add your dialog message here. Animated dialog description place.")
        .withDialogColor("#FFFFFF")
    .withButton1Text("OK")
        .withButton2Text("Cancel")
    .withDuration(700)
    .withEffect(Effectstype.Fall)
        .show();
}

public void animatedDialog2(View view) {
    materialDesignAnimatedDialog
    .withTitle("Animated Flip Dialog Title")
        .withMessage("Add your dialog message here. Animated dialog description place.")
        .withDialogColor("#1c90ec")
    .withButton1Text("OK")
        .withButton2Text("Cancel")
    .withDuration(700)
    .withEffect(Effectstype.Flip)
        .show();
}

public void animatedDialog3(View view) {
    materialDesignAnimatedDialog
    .withTitle("Animated Shake Dialog Title")
        .withMessage("Add your dialog message here. Animated dialog description place.")
        .withDialogColor("#1c90ec")
    .withButton1Text("OK")
        .withButton2Text("Cancel")
    .withDuration(700)
    .withEffect(Effectstype.Shake)
        .show();
}

public void animatedDialog4(View view) {
    materialDesignAnimatedDialog
    .withTitle("Animated Slide Top Dialog Title")
        .withMessage("Add your dialog message here. Animated dialog description place.")
        .withDialogColor("#1c90ec")
    .withButton1Text("OK")
        .withButton2Text("Cancel")
    .withDuration(700)
    .withEffect(Effectstype.Slidetop)
        .show();
}

```

Add the below lines in your build.gradle to include the NifyBuilder(CustomView)

build.gradle

```
dependencies {
```

```
compile 'com.nineoldandroids:library:2.4.0'  
compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'  
}
```

参考链接 : <https://github.com/sd6352051/NiftyDialogEffects>

```
compile 'com.nineoldandroids:library:2.4.0'  
compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'  
}
```

Reference Link : <https://github.com/sd6352051/NiftyDialogEffects>

第65章：GreenDAO

GreenDAO是一个对象关系映射（ORM）库，帮助开发者使用SQLite数据库进行持久化本地存储。

第65.1节：用于SELECT、INSERT、DELETE、UPDATE查询的辅助方法

本示例展示了一个辅助类，包含执行数据查询时有用的方法。这里的每个方法都使用了Java泛型，以实现高度灵活性。

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                         WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                               WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
```

Chapter 65: GreenDAO

GreenDAO is an Object-Relational Mapping library to help developers use SQLite databases for persistent local storage.

Section 65.1: Helper methods for SELECT, INSERT, DELETE, UPDATE queries

This example shows a helper class that contains methods useful, when executing the queries for data. Every method here uses Java Generic's in order to be very flexible.

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                         WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                               WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
```

```

QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
    Property sortProperty,
    String sortStrategy,
WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T, ?> absDao,
    Property sortProperty,
    boolean handleNulls,
    String sortStrategy,
    WhereCondition... 条件) {
    if (!处理空值) {
        return 按条件和排序选择元素(absDao, sortProperty, sortStrategy, conditions);
    }
    if (absDao == null) {
        return null;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1 ELSE 0 END), " +
    "T." + sortProperty.columnName + " " + sortStrategy);
List<T> items = qb.list();
    return items != null ? items : null;
}

public <T, V extends Class> List<T> 按连接选择(AbstractDao<T, ?> absDao,
    V className,
    Property property, WhereCondition whereCondition)
{
    QueryBuilder<T> qb = absDao.queryBuilder();
        qb.join(className, property).where(whereCondition);
    return qb.list();
}

public <T> void 按条件删除元素(AbstractDao<T, ?> absDao,
    WhereCondition... 条件) {
    if (absDao == null) {
        return;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : 条件) {
        qb = qb.where(condition);
    }
}

```

```

QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
    Property sortProperty,
    String sortStrategy,
    WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T, ?> absDao,
    Property sortProperty,
    boolean handleNulls,
    String sortStrategy,
    WhereCondition... conditions) {
    if (!handleNulls) {
        return selectElementsByConditionAndSort(absDao, sortProperty, sortStrategy, conditions);
    }
    if (absDao == null) {
        return null;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1 ELSE 0 END), " +
    "T." + sortProperty.columnName + " " + sortStrategy);
List<T> items = qb.list();
    return items != null ? items : null;
}

public <T, V extends Class> List<T> selectByJoin(AbstractDao<T, ?> absDao,
    V className,
    Property property, WhereCondition whereCondition)
{
    QueryBuilder<T> qb = absDao.queryBuilder();
        qb.join(className, property).where(whereCondition);
    return qb.list();
}

public <T> void deleteElementsByCondition(AbstractDao<T, ?> absDao,
    WhereCondition... conditions) {
    if (absDao == null) {
        return;
    }
QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
}

```

```

        }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object) {
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                              V className,
                                              Property property, WhereCondition whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}

```

```

        }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object) {
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                              V className,
                                              Property property, WhereCondition whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}

```

第65.2节：使用GreenDAO 3.X创建具有复合主键的实体

当为具有复合主键的表创建模型时，需要对模型实体的对象进行额外处理以满足这些约束。

下面的示例SQL表和实体演示了存储客户对在线商店中商品的评价的结构。在此示例中，我们希望customer_id和item_id列作为复合主键，允许特定客户和商品之间只存在一条评价。

SQL表

```

CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);

```

通常我们在实体类中相应字段上使用@Id和@Unique注解，但对于复合主键，我们需要进行如下操作：

Section 65.2: Creating an Entity with GreenDAO 3.X that has a Composite Primary Key

When creating a model for a table that has a composite primary key, additional work is required on the Object for the model Entity to respect those constraints.

The following example SQL table and Entity demonstrates the structure to store a review left by a customer for an item in an online store. In this example, we want the customer_id and item_id columns to be a composite primary key, allowing only one review to exist between a specific customer and item.

SQL Table

```

CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);

```

Usually we would use the @Id and @Unique annotations above the respective fields in the entity class, however for a composite primary key we do the following:

- 在类级别的@Entity注解内添加@Index注解。value属性包含一个逗号-由组成键的字段组成的分隔列表。使用如示例所示的unique属性来强制键的唯一性。
- GreenDAO要求每个实体都有一个long或Long对象作为主键。我们仍然需要将此添加到实体类，但我们不需要使用它或担心它会影响我们的实现。在下面的示例中，它被称为localID

实体

```
@Entity(indexes = { @Index(value = "customer_id,item_id", unique = true)})
public class Review {

    @Id(autoincrement = true)
    private Long localID;

    private String customer_id;
    private String item_id;

    @NotNull
    private Integer star_rating;

    private String content;

    public Review() {}
}
```

第65.3节：开始使用GreenDao v3.X

添加GreenDao库依赖和Gradle插件后，我们需要先创建一个实体对象。

实体

实体是一个普通的Java对象(POJO)，用于建模数据库中的某些数据。GreenDao将使用此类在SQLite数据库中创建表，并自动生成辅助类，供我们访问和存储数据而无需编写SQL语句。

```
@Entity
public class Users {

    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // 字段的getter和setter方法...
}
```

GreenDao的一次性设置

每次应用启动时都需要初始化GreenDao。GreenDao建议将这段代码放在Application类中或其他只会运行一次的位置。

- Add the @Index annotation inside the class-level @Entity annotation. The value property contains a comma-delimited list of the fields that make up the key. Use the unique property as shown to enforce uniqueness on the key.
- GreenDAO requires every Entity have a long or Long object as a primary key. We still need to add this to the Entity class, however we do not need to use it or worry about it affecting our implementation. In the example below it is called localID

Entity

```
@Entity(indexes = { @Index(value = "customer_id,item_id", unique = true)})
public class Review {

    @Id(autoincrement = true)
    private Long localID;

    private String customer_id;
    private String item_id;

    @NotNull
    private Integer star_rating;

    private String content;

    public Review() {}
}
```

Section 65.3: Getting started with GreenDao v3.X

After adding the GreenDao library dependency and Gradle plugin, we need to first create an entity object.

Entity

An entity is a Plain Old Java Object (POJO) that models some data in the database. GreenDao will use this class to create a table in the SQLite database and automatically generate helper classes we can use to access and store data without having to write SQL statements.

```
@Entity
public class Users {

    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // Getters and setters for the fields...
}
```

One-time GreenDao setup

Each time an application is launched GreenDao needs to be initialized. GreenDao suggests keeping this code in an Application class or somewhere it will only be run once.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

GreenDao 辅助类

实体对象创建后，GreenDao 会自动创建用于与数据库交互的辅助类。这些类的命名与创建的实体对象名称相似，后缀为Dao，并且从daoSession对象中获取。

```
UsersDao usersDao = daoSession.getUsersDao();
```

现在可以使用该 Dao 对象和实体对象执行许多典型的数据库操作。

查询

```
String email = "jdoe@example.com";
String firstname = "John";

// 单个用户查询，WHERE 邮箱匹配 "jdoe@example.com"
Users user = userDao.queryBuilder()
    .where(Properties.Email.eq(email))
    .unique();

// 多用户查询 WHERE firstname = "John"
List<Users> user = userDao.queryBuilder()
    .where(Properties.Firstname.eq(firstname))
    .list();
```

插入

```
Users newUser = new User("John", "Doe", "jdoe@example.com");
usersDao.insert(newUser);
```

更新

```
// 修改之前检索到的用户对象并更新
user.setLastname("Dole");
usersDao.update(user);
```

删除

```
// 删除之前检索到的用户对象
usersDao.delete(user);
```

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

GreenDao Helper Classes

After the entity object is created, GreenDao automatically creates the helper classes used to interact with the database. These are named similarly to the name of the entity object that was created, followed by Dao and are retrieved from the daoSession object.

```
UsersDao usersDao = daoSession.getUsersDao();
```

Many typical database actions can now be performed using this Dao object with the entity object.

Query

```
String email = "jdoe@example.com";
String firstname = "John";

// Single user query WHERE email matches "jdoe@example.com"
Users user = userDao.queryBuilder()
    .where(Properties.Email.eq(email))
    .unique();

// Multiple user query WHERE firstname = "John"
List<Users> user = userDao.queryBuilder()
    .where(Properties.Firstname.eq(firstname))
    .list();
```

Insert

```
Users newUser = new User("John", "Doe", "jdoe@example.com");
usersDao.insert(newUser);
```

Update

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Delete

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

第66章：工具属性

第66.1节：设计时布局属性

这些属性在Android Studio中渲染布局时使用，但对运行时没有影响。

通常你可以使用任何Android框架属性，只需在布局预览时使用tools:命名空间，而不是android:命名空间。你可以同时添加android:命名空间属性（运行时使用）和对应的tools:属性（仅在布局预览中覆盖运行时属性）。

只需按照备注部分描述定义tools命名空间即可。

例如text属性：

```
<EditText  
    tools:text="My Text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

或者使用visibility属性来取消预览视图：

```
<LinearLayout  
    android:id="@+id/l11"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    tools:visibility="gone" />
```

或者使用context属性将布局与活动或片段关联

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    tools:context=".MainActivity" >
```

或者使用showIn属性在另一个布局中查看并包含布局预览

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/text"  
    tools:showIn="@layout/activity_main" />
```

Chapter 66: Tools Attributes

Section 66.1: Designtime Layout Attributes

These attributes are used when the layout is rendered in Android Studio, but have no impact on the runtime.

In general you can use any Android framework attribute, just using the tools: namespace rather than the android: namespace for layout preview. You can add both the android: namespace attribute (which is used at runtime) and the matching tools: attribute (which overrides the runtime attribute in the layout preview only).

Just define the tools namespace as described in the remarks section.

For example the text attribute:

```
<EditText  
    tools:text="My Text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

Or the visibility attribute to unset a view for preview:

```
<LinearLayout  
    android:id="@+id/l11"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    tools:visibility="gone" />
```

Or the context attribute to associate the layout with activity or fragment

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    tools:context=".MainActivity" >
```

Or the showIn attribute to see and included layout preview in another layout

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/text"  
    tools:showIn="@layout/activity_main" />
```

第67章：格式化字符串

第67.1节：格式化字符串资源

您可以在字符串资源中添加通配符，并在运行时填充它们：

1. 编辑 strings.xml

```
<string name="my_string">这是 %1$s</string>
```

2. 根据需要格式化字符串

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

第67.2节：将数据类型格式化为字符串及反向操作

数据类型到字符串的格式化

像 int、float、double、long、boolean 这样的数据类型可以使用 String.valueOf() 格式化为字符串。

```
String.valueOf(1); //输出 -> "1"
String.valueOf(1.0); //输出 -> "1.0"
String.valueOf(1.2345); //输出 -> "1.2345"
String.valueOf(true); //输出 -> "true"
```

相反，将字符串格式化为其他数据类型

```
Integer.parseInt("1"); //输出 -> 1
Float.parseFloat("1.2"); //输出 -> 1.2
Boolean.parseBoolean("true"); //输出 -> true
```

第67.3节：将时间戳格式化为字符串

有关模式的完整说明，请参见 [SimpleDateFormat参考文档](#)

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Chapter 67: Formatting Strings

Section 67.1: Format a string resource

You can add wildcards in string resources and populate them at runtime:

1. Edit strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. Format string as needed

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Section 67.2: Formatting data types to String and vice versa

Data types to string formatting

Data types like int, float, double, long, boolean can be formatted to string using String.valueOf().

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

Vise versa of this, formatting string to other data type

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Section 67.3: Format a timestamp to string

For full description of patterns, see [SimpleDateFormat reference](#)

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

第68章：SpannableString

第68.1节：为TextView添加样式

在以下示例中，我们创建一个活动来显示单个文本视图。

TextView 将使用一个SpannableString作为其内容，这将展示一些可用的样式。

我们将对文本做以下操作：

- 放大
- 加粗
- 下划线
- 斜体
- 删除线
- 着色
- 高亮
- 显示为上标
- 显示为下标
- 显示为链接
- 设置为可点击。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
  
SpannableString styledString= new  
SpannableString("Large" // 索引 0 - 5  
+ "Bold" // 索引 7 - 11  
+ "Underlined" // 索引 13 - 23  
+ "Italic" // 索引 25 - 31  
+ "Strikethrough" // 索引 33 - 46  
+ "彩色" // 索引 48 - 55  
+ "高亮" // 索引 57 - 68  
+ "K 上标" // "上标" 索引 72 - 83  
+ "K 下标" // "下标" 索引 87 - 96  
+ "网址" // 索引 98 - 101  
+ "可点击"); // 索引 103 - 112  
  
// 使文本变为原来的两倍大  
styledString.setSpan(new RelativeSizeSpan(2f), 0, 5, 0);  
  
// 使文本加粗  
styledString.setSpan(new StyleSpan(Typeface.BOLD), 7, 11, 0);  
  
// 下划线文本  
styledString.setSpan(new UnderlineSpan(), 13, 23, 0);  
  
// 使文本变为斜体  
styledString.setSpan(new StyleSpan(Typeface.ITALIC), 25, 31, 0);  
  
styledString.setSpan(new StrikethroughSpan(), 33, 46, 0);  
  
// 更改文本颜色  
styledString.setSpan(new ForegroundColorSpan(Color.GREEN), 48, 55, 0);  
  
// 高亮文本  
styledString.setSpan(new BackgroundColorSpan(Color.CYAN), 57, 68, 0);
```

Chapter 68: SpannableString

Section 68.1: Add styles to a TextView

In the following example, we create an Activity to display a single TextView.

The TextView will use a SpannableString as its content, which will illustrate some of the available styles.

Here's what we're gonna do with the text:

- Make it larger
- Bold
- Underline
- Italicize
- Strike-through
- Colored
- Highlighted
- Show as superscript
- Show as subscript
- Show as a link
- Make it clickable.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
  
SpannableString styledString  
= new SpannableString("Large\n\n" // index 0 - 5  
+ "Bold\n\n" // index 7 - 11  
+ "Underlined\n\n" // index 13 - 23  
+ "Italic\n\n" // index 25 - 31  
+ "Strikethrough\n\n" // index 33 - 46  
+ "Colored\n\n" // index 48 - 55  
+ "Highlighted\n\n" // index 57 - 68  
+ "K Superscript\n\n" // "Superscript" index 72 - 83  
+ "K Subscript\n\n" // "Subscript" index 87 - 96  
+ "Url\n\n" // index 98 - 101  
+ "Clickable\n\n"); // index 103 - 112  
  
// make the text twice as large  
styledString.setSpan(new RelativeSizeSpan(2f), 0, 5, 0);  
  
// make text bold  
styledString.setSpan(new StyleSpan(Typeface.BOLD), 7, 11, 0);  
  
// underline text  
styledString.setSpan(new UnderlineSpan(), 13, 23, 0);  
  
// make text italic  
styledString.setSpan(new StyleSpan(Typeface.ITALIC), 25, 31, 0);  
  
styledString.setSpan(new StrikethroughSpan(), 33, 46, 0);  
  
// change text color  
styledString.setSpan(new ForegroundColorSpan(Color.GREEN), 48, 55, 0);  
  
// highlight text  
styledString.setSpan(new BackgroundColorSpan(Color.CYAN), 57, 68, 0);
```

```

// 上标
styledString.setSpan(new SuperscriptSpan(), 72, 83, 0);
// 使上标文字变小
styledString.setSpan(new RelativeSizeSpan(0.5f), 72, 83, 0);

// 下标
styledString.setSpan(new SubscriptSpan(), 87, 96, 0);
// 使下标文字变小
styledString.setSpan(new RelativeSizeSpan(0.5f), 87, 96, 0);

// 链接
styledString.setSpan(new URLSpan("http://www.google.com"), 98, 101, 0);

// 可点击文字
ClickableSpan clickableSpan = new ClickableSpan() {
    @Override
    public void onClick(View widget) {
        // 我们显示一个Toast。你可以在这里做任何你想做的事情。
        Toast.makeText(SpanExample.this, "Clicked", Toast.LENGTH_SHORT).show();
    }
};

styledString.setSpan(clickableSpan, 103, 112, 0);

// 将样式字符串赋给TextView
TextView textView = new TextView(this);

// 这一步是url和可点击样式所必需的。
textView.setMovementMethod(LinkMovementMethod.getInstance());

// 使其整洁
textView.setGravity(Gravity.CENTER);
textView.setBackgroundColor(Color.WHITE);

textView.setText(styledString);

setContentView(textView);
}

```

```

// superscript
styledString.setSpan(new SuperscriptSpan(), 72, 83, 0);
// make the superscript text smaller
styledString.setSpan(new RelativeSizeSpan(0.5f), 72, 83, 0);

// subscript
styledString.setSpan(new SubscriptSpan(), 87, 96, 0);
// make the subscript text smaller
styledString.setSpan(new RelativeSizeSpan(0.5f), 87, 96, 0);

// url
styledString.setSpan(new URLSpan("http://www.google.com"), 98, 101, 0);

// clickable text
ClickableSpan clickableSpan = new ClickableSpan() {
    @Override
    public void onClick(View widget) {
        // We display a Toast. You could do anything you want here.
        Toast.makeText(SpanExample.this, "Clicked", Toast.LENGTH_SHORT).show();
    }
};

styledString.setSpan(clickableSpan, 103, 112, 0);

// Give the styled string to a TextView
TextView textView = new TextView(this);

// this step is mandated for the url and clickable styles.
textView.setMovementMethod(LinkMovementMethod.getInstance());

// make it neat
textView.setGravity(Gravity.CENTER);
textView.setBackgroundColor(Color.WHITE);

textView.setText(styledString);

setContentView(textView);
}

```



Large

Bold

Underlined

Italic

~~Strikethrough~~

Colored

Highlighted

K^{Superscript}

K_{Subscript}

Url

Clickable



Large

Bold

Underlined

Italic

~~Strikethrough~~

Colored

Highlighted

K^{Superscript}

K_{Subscript}

Url

Clickable

结果将如下所示：

第68.2节：多字符串，多颜色

方法：`setSpanColor`

```
public Spanned setSpanColor(String string, int color){  
    SpannableStringBuilder builder = new SpannableStringBuilder();  
    SpannableString ss = new SpannableString(string);  
    ss.setSpan(new ForegroundColorSpan(color), 0, string.length(), 0);  
    builder.append(ss);  
    return ss;  
}
```

And the result will look like this:

Section 68.2: Multi string , with multi color

Method: `setSpanColor`

```
public Spanned setSpanColor(String string, int color){  
    SpannableStringBuilder builder = new SpannableStringBuilder();  
    SpannableString ss = new SpannableString(string);  
    ss.setSpan(new ForegroundColorSpan(color), 0, string.length(), 0);  
    builder.append(ss);  
    return ss;  
}
```

用法：

```
String a = getString(R.string.string1);
String b = getString(R.string.string2);

Spanned color1 = setSpanColor(a, Color.CYAN);
Spanned color2 = setSpanColor(b, Color.RED);
Spanned mixedColor = TextUtils.concat(color1, " ", color2);
// 现在我们使用 `mixedColor`
```

Usage:

```
String a = getString(R.string.string1);
String b = getString(R.string.string2);

Spanned color1 = setSpanColor(a, Color.CYAN);
Spanned color2 = setSpanColor(b, Color.RED);
Spanned mixedColor = TextUtils.concat(color1, " ", color2);
// Now we use `mixedColor`
```

第69章：通知

第69.1节：适用于旧设备的带有滚动字幕的悬浮通知

以下是如何为支持的设备创建悬浮通知，并为旧设备使用滚动字幕的方法。

```
// 点击通知将打开MainActivity
Intent i = new Intent(this, MainActivity.class);

// 之后使用的一个动作
// 定义为应用常量：
// public static final String MESSAGE_CONSTANT = "com.example.myapp.notification";
i.setAction(MainActivity.MESSAGE_CONSTANT);
// 你也可以使用额外参数
i.putExtra("some_extra", "testValue");

i.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_SINGLE_TOP);
PendingIntent notificationIntent = PendingIntent.getActivity(this, 999, i,
PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this.getApplicationContext());
builder.setContentIntent(notificationIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(this.getResources(),
android.R.drawable.ic_menu_view));
builder.setSmallIcon(android.R.drawable.ic_dialog_map);
builder.setContentText("测试消息文本");
builder.setTicker("测试滚动文本");
builder.setContentTitle("测试消息标题");

// 设置高优先级以显示悬浮通知
builder.setPriority(NotificationCompat.PRIORITY_HIGH);
builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);

// 除非播放声音，否则不会显示“悬浮通知”
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);

NotificationManager mNotificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(999, builder.build());
```

以下是在搭载Heads Up通知的Android Marshmallow上的显示效果：

Chapter 69: Notifications

Section 69.1: Heads Up Notification with Ticker for older devices

Here is how to make a Heads Up Notification for capable devices, and use a Ticker for older devices.

```
// Tapping the Notification will open up MainActivity
Intent i = new Intent(this, MainActivity.class);

// an action to use later
// defined as an app constant:
// public static final String MESSAGE_CONSTANT = "com.example.myapp.notification";
i.setAction(MainActivity.MESSAGE_CONSTANT);
// you can use extras as well
i.putExtra("some_extra", "testValue");

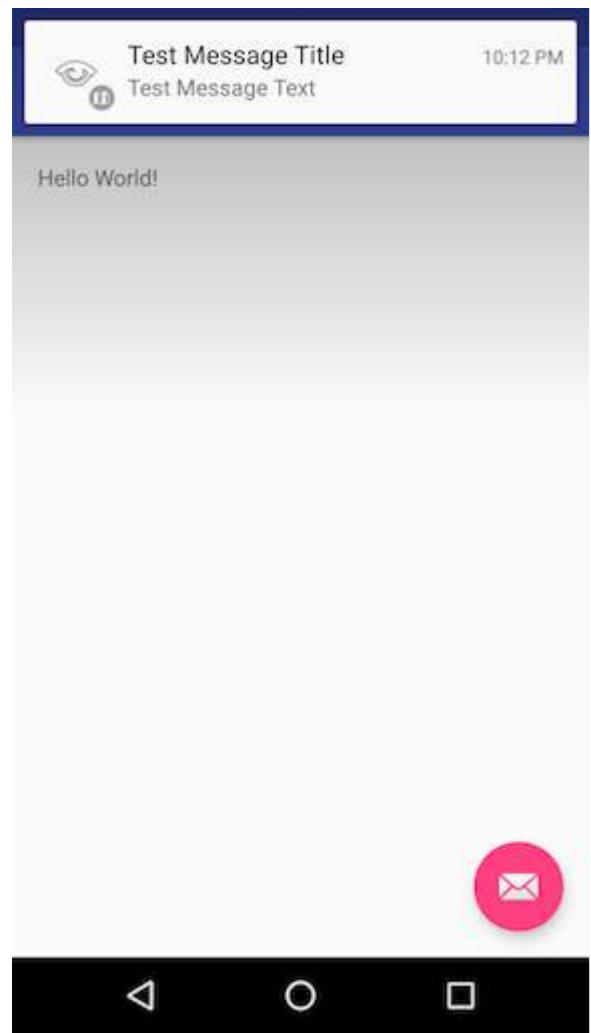
i.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_SINGLE_TOP);
PendingIntent notificationIntent = PendingIntent.getActivity(this, 999, i,
PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this.getApplicationContext());
builder.setContentIntent(notificationIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(this.getResources(),
android.R.drawable.ic_menu_view));
builder.setSmallIcon(android.R.drawable.ic_dialog_map);
builder.setContentText("Test Message Text");
builder.setTicker("Test Ticker Text");
builder.setContentTitle("Test Message Title");

// set high priority for Heads Up Notification
builder.setPriority(NotificationCompat.PRIORITY_HIGH);
builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);

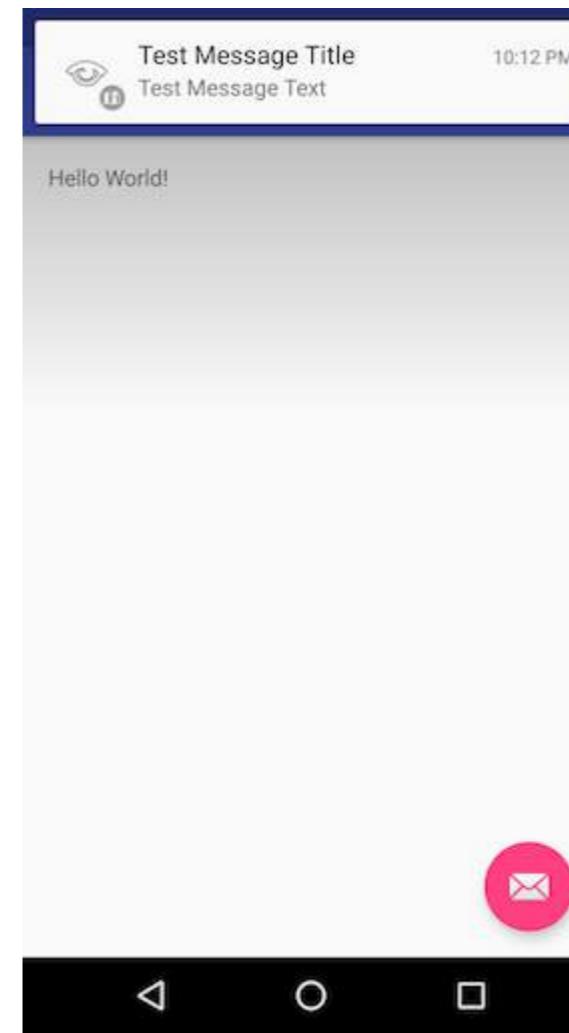
// It won't show "Heads Up" unless it plays a sound
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);

NotificationManager mNotificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(999, builder.build());
```

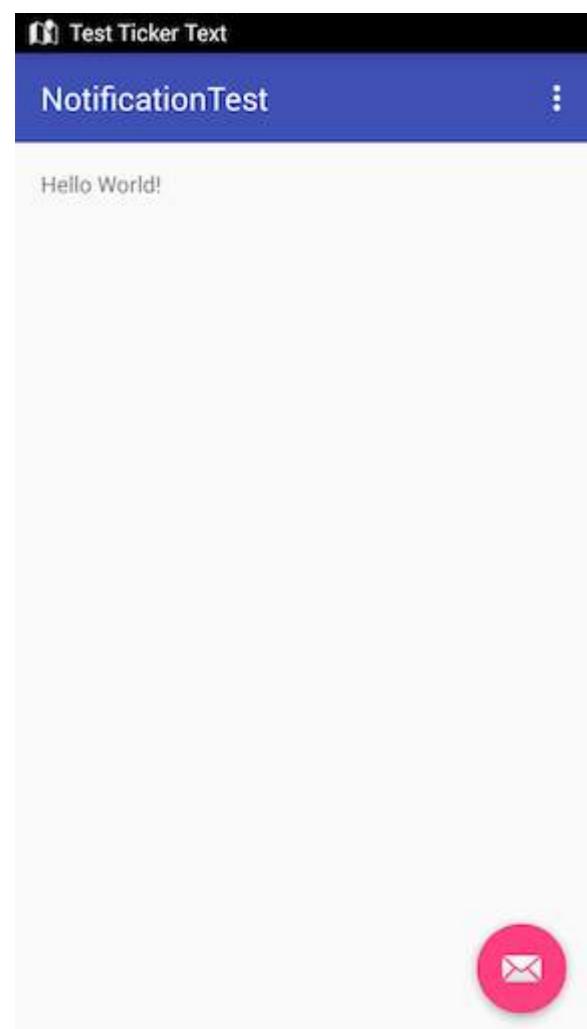
Here is what it looks like on Android Marshmallow with the Heads Up Notification:



以下是在搭载Ticker的Android KitKat上的显示效果：

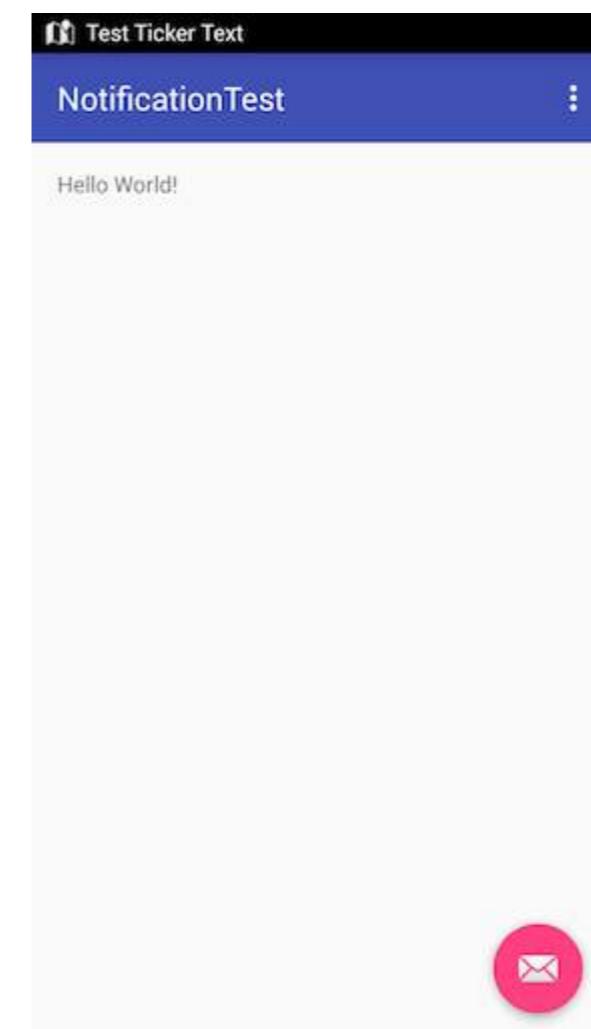


Here is what it looks like on Android KitKat with the Ticker:



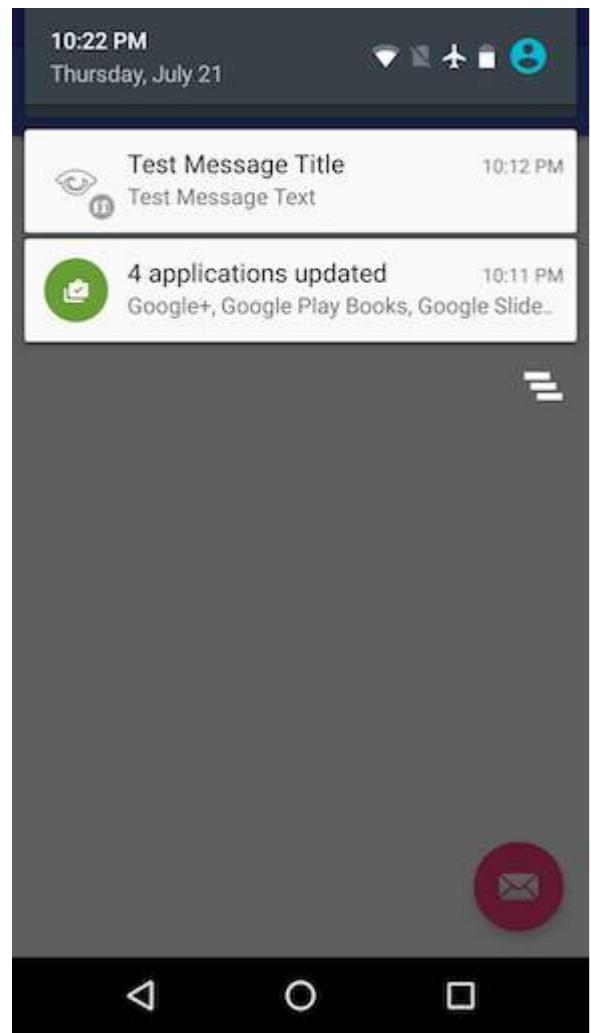
在所有Android版本中，通知都会显示在通知栏中。

Android 6.0 Marshmallow :

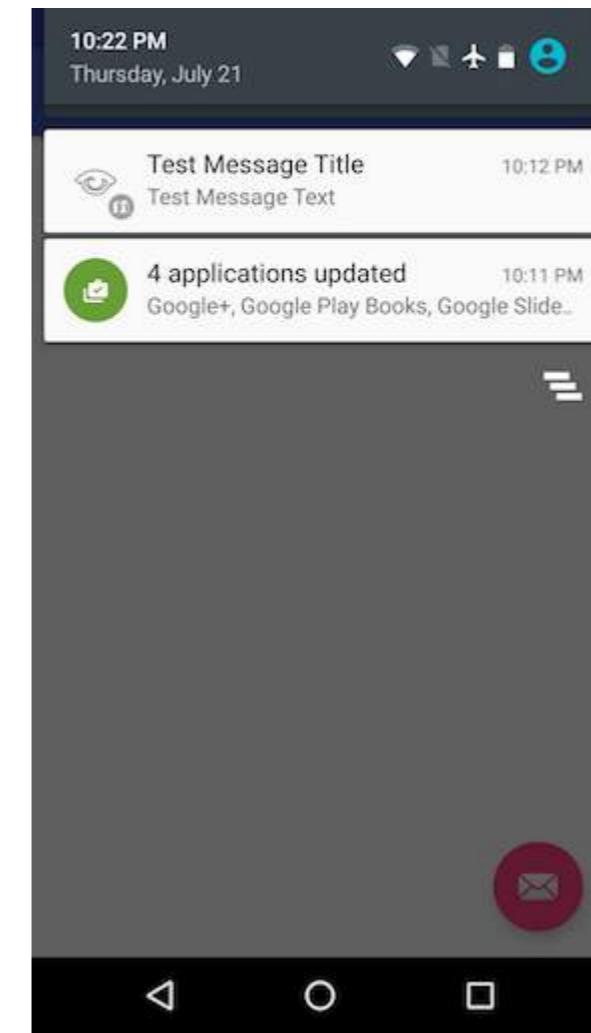


On all Android versions, the Notification is shown in the notification drawer.

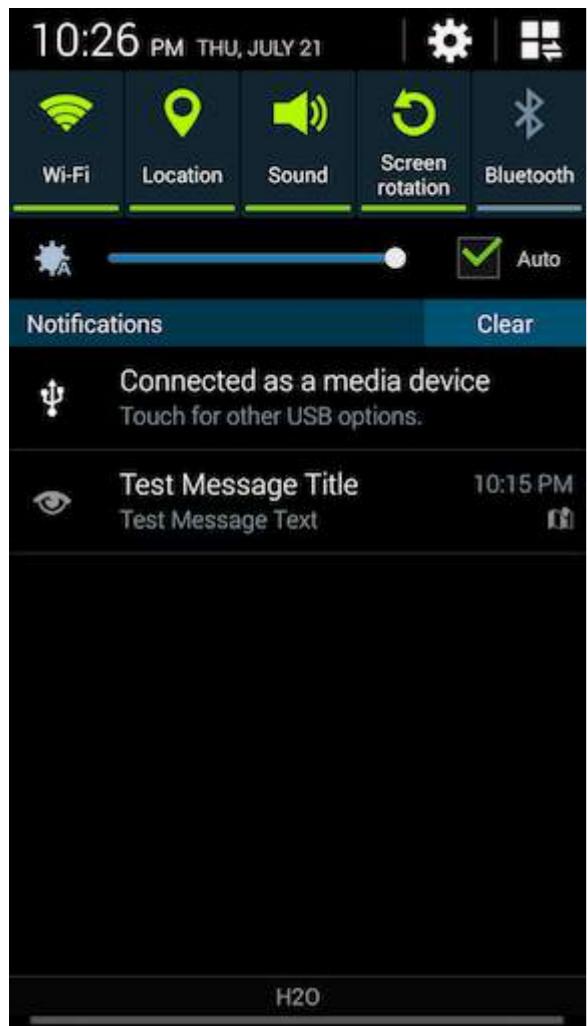
Android 6.0 Marshmallow:



Android 4.4.x KitKat :



Android 4.4.x KitKat:



第69.2节：创建一个简单的通知

此示例展示了如何创建一个简单的通知，当用户点击时启动一个应用程序。

指定通知的内容：

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.ic_launcher) // 通知图标
    .setContentTitle("简单通知") // 标题
    .setContentText("Hello word") // 正文消息
    .setAutoCancel(true); // 点击后清除通知
```

创建点击时触发的意图：

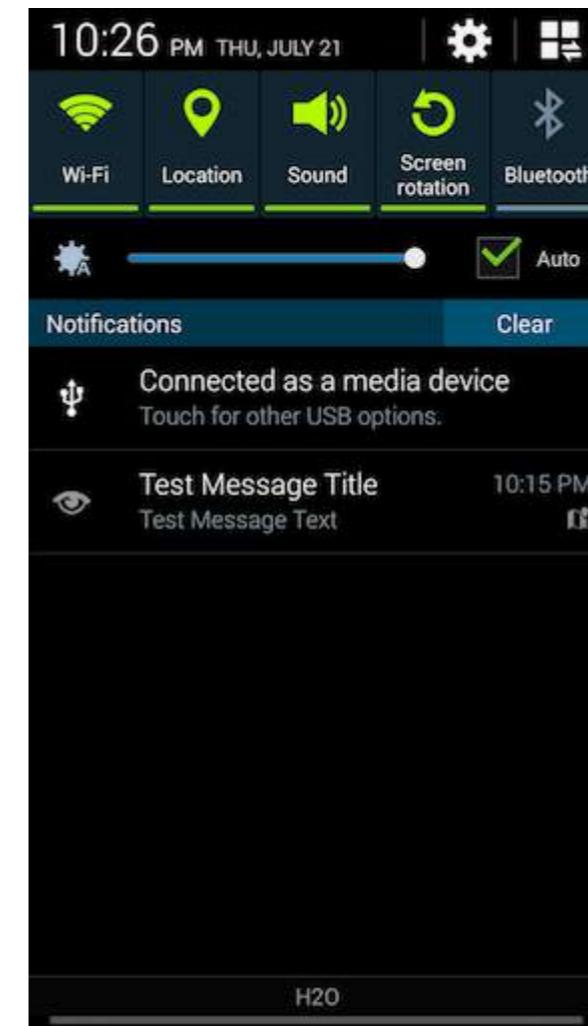
```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

最后，构建通知并显示

```
NotificationManager mNotificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

第69.3节：设置自定义通知 - 显示完整内容文本

如果您想在上下文中显示较长的文本，需要设置自定义内容。



Section 69.2: Creating a simple Notification

This example shows how to create a simple notification that starts an application when the user clicks it.

Specify the notification's content:

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.ic_launcher) // notification icon
    .setContentTitle("Simple notification") // title
    .setContentText("Hello word") // body message
    .setAutoCancel(true); // clear notification when clicked
```

Create the intent to fire on click:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

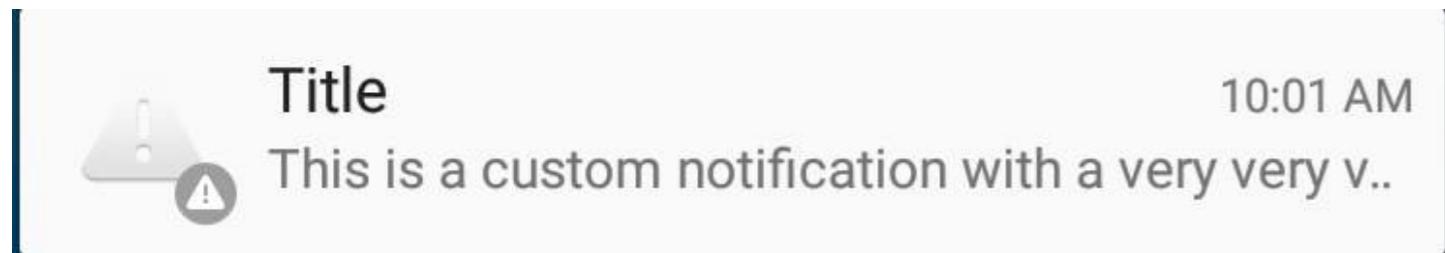
Finally, build the notification and show it

```
NotificationManager mNotificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

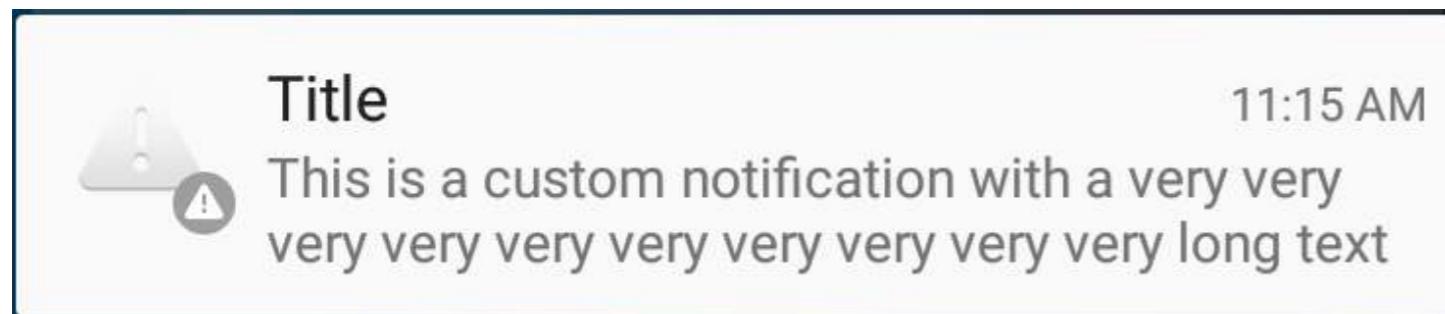
Section 69.3: Set custom notification - show full content text

If you want have a long text to display in the context, you need to set a custom content.

例如，你有以下内容：



但你希望你的文本能够完全显示：



你所需要做的，就是给你的内容添加一个样式，如下所示：

```
private void generateNotification(Context context) {
    String message = "这是一个自定义通知，包含非常非常非常非常非常非常非常非常非常非常长的文本";
    Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
        android.R.drawable.ic_dialog_alert);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

    builder.setContentTitle("标题").setContentText(message)
        .setSmallIcon(android.R.drawable.ic_dialog_alert)
        .setLargeIcon(largeIcon)
        .setAutoCancel(true)
        .setWhen(System.currentTimeMillis())
        .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

    Notification notification = builder.build();
    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(context);
    notificationManager.notify(101, notification);
}
```

第69.4节：动态获取大图标的正确像素大小

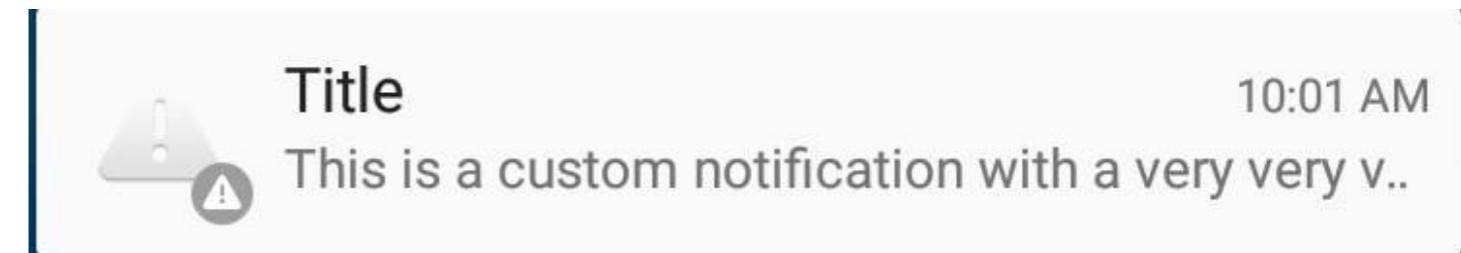
如果您正在创建图像、解码图像或调整图像大小以适应大型通知图像区域，可以这样获取正确的像素尺寸：

```
Resources resources = context.getResources();
int width = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_width);
int height = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_height);
```

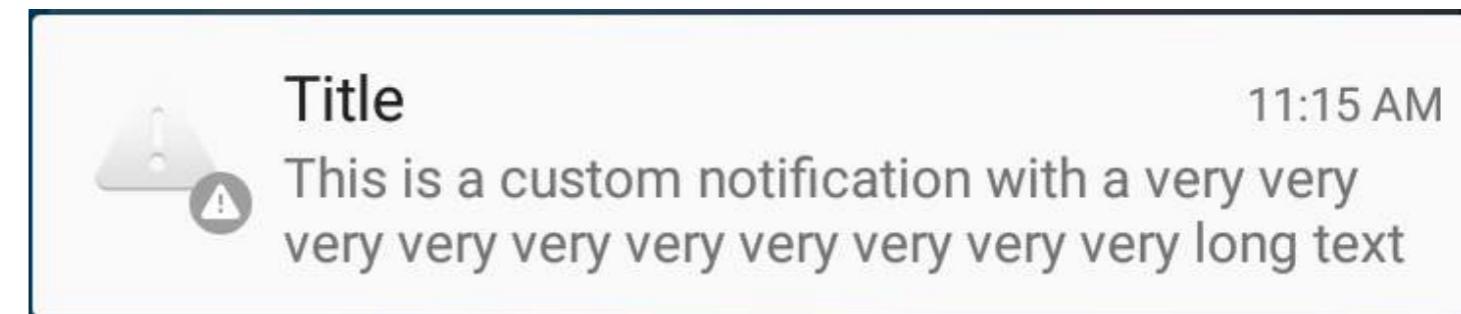
第69.5节：带操作按钮的持续通知

```
// 取消具有相同ID的旧通知,
NotificationManager notificationMgr =
```

For example, you have this:



But you wish your text will be fully shown:



All you need to do, is to **add a style** to your content like below:

```
private void generateNotification(Context context) {
    String message = "This is a custom notification with a very very very very very very very very very long text";
    Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
        android.R.drawable.ic_dialog_alert);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

    builder.setContentTitle("Title").setContentText(message)
        .setSmallIcon(android.R.drawable.ic_dialog_alert)
        .setLargeIcon(largeIcon)
        .setAutoCancel(true)
        .setWhen(System.currentTimeMillis())
        .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

    Notification notification = builder.build();
    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(context);
    notificationManager.notify(101, notification);
}
```

Section 69.4: Dynamically getting the correct pixel size for the large icon

If you're creating an image, decoding an image, or resizing an image to fit the large notification image area, you can get the correct pixel dimensions like so:

```
Resources resources = context.getResources();
int width = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_width);
int height = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_height);
```

Section 69.5: Ongoing notification with Action button

```
// Cancel older notification with same id,
NotificationManager notificationMgr =
```

```

(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationMgr.cancel(CALL_NOTIFY_ID); // 任意常量值

// 创建待定意图,
Intent notificationIntent = null;
PendingIntent contentIntent = null;
notificationIntent = new Intent(context, YourActivityName);
contentIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

// 通知构建器
builder = new NotificationCompat.Builder(context);
builder.setContentText("进行中的通知..");
builder.setContentTitle("进行中通知示例");
builder.setSmallIcon(R.drawable.notification_icon);
builder.setUsesChronometer(true);
builder.setDefaults(Notification.DEFAULT_LIGHTS);
builder.setContentIntent(contentIntent);
builder.setOngoing(true);

// 在通知中添加操作按钮
Intent intent = new Intent("action.name");
PendingIntent pIntent = PendingIntent.getBroadcast(context, 1, intent, 0);
builder.addAction(R.drawable.action_button_icon, "操作按钮名称", pIntent);

// 使用 notificationMgr 发送通知
Notification finalNotification = builder.build();
notificationMgr.notify(CALL_NOTIFY_ID, finalNotification);

```

注册一个广播接收器以处理操作按钮点击事件，监听相同的 action。

第69.6节：设置通知的不同优先级

```

NotificationCompat.Builder mBuilder =
    (NotificationCompat.Builder) new NotificationCompat.Builder(context)
        .setSmallIcon(R.drawable.some_small_icon)
        .setContentTitle("标题")
        .setContentText("这是一个具有最大优先级的测试通知")
        .setPriority(Notification.PRIORITY_MAX);

```

当通知包含图片且您希望在收到通知时自动展开图片时，使用 "PRIORITY_MAX"，您也可以根据需求使用其他优先级级别

不同优先级级别信息：

PRIORITY_MAX -- 用于关键且紧急的通知，提醒用户某个时间紧迫的情况，需要在继续执行特定任务之前解决。

PRIORITY_HIGH -- 主要用于重要的通信，如消息或聊天事件，内容对用户特别有吸引力。高优先级通知会触发悬浮通知显示。

PRIORITY_DEFAULT -- 用于所有不属于上述其他优先级的通知。

PRIORITY_LOW -- 用于您希望用户知晓但不那么紧急的通知。低优先级通知通常显示在列表底部，适合用于例如

```

(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationMgr.cancel(CALL_NOTIFY_ID); // any constant value

```

```

// Create Pending Intent,
Intent notificationIntent = null;
PendingIntent contentIntent = null;
notificationIntent = new Intent(context, YourActivityName);
contentIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

// Notification builder
builder = new NotificationCompat.Builder(context);
builder.setContentText("Ongoing Notification..");
builder.setContentTitle("ongoing notification sample");
builder.setSmallIcon(R.drawable.notification_icon);
builder.setUsesChronometer(true);
builder.setDefaults(Notification.DEFAULT_LIGHTS);
builder.setContentIntent(contentIntent);
builder.setOngoing(true);

```

```

// Add action button in the notification
Intent intent = new Intent("action.name");
PendingIntent pIntent = PendingIntent.getBroadcast(context, 1, intent, 0);
builder.addAction(R.drawable.action_button_icon, "Action button name", pIntent);

```

```

// Notify using notificationMgr
Notification finalNotification = builder.build();
notificationMgr.notify(CALL_NOTIFY_ID, finalNotification);

```

Register a broadcast receiver for the same action to handle action button click event.

Section 69.6: Setting Different priorities in notification

```

NotificationCompat.Builder mBuilder =
    (NotificationCompat.Builder) new NotificationCompat.Builder(context)
        .setSmallIcon(R.drawable.some_small_icon)
        .setContentTitle("Title")
        .setContentText("This is a test notification with MAX priority")
        .setPriority(Notification.PRIORITY_MAX);

```

When notification contains image and you want to auto expand image when notification received use "PRIORITY_MAX"，you can use other priority levels as per requirements

Different Priority Levels Info:

PRIORITY_MAX -- Use for critical and urgent notifications that alert the user to a condition that is time-critical or needs to be resolved before they can continue with a particular task.

PRIORITY_HIGH -- Use primarily for important communication, such as message or chat events with content that is particularly interesting for the user. High-priority notifications trigger the heads-up notification display.

PRIORITY_DEFAULT -- Use for all notifications that don't fall into any of the other priorities described here.

PRIORITY_LOW -- Use for notifications that you want the user to be informed about, but that are less urgent. Low-priority notifications tend to show up at the bottom of the list, which makes them a good choice for things like

公开或非定向的社交更新：用户已请求接收此类通知，但这些通知绝不应优先于紧急或直接的通信。

PRIORITY_MIN -- 用于上下文或后台信息，如天气信息或上下文位置信息。最低优先级通知不会出现在状态栏中。用户通过展开通知栏来发现它们。

参考资料：[Material Design 指南 - 通知](#)

第69.7节：使用`Picasso`库设置自定义通知图标

```
PendingIntent pendingIntent = PendingIntent.getActivity(context,
    uniqueIntentId, intent, PendingIntent.FLAG_CANCEL_CURRENT);

final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.remote_view_notification);
remoteViews.setImageResource(R.id.remoteview_notification_icon,
R.mipmap.ic_navigation_favorites);

Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder =
    new NotificationCompat.Builder(context)
.setSmallIcon(R.mipmap.ic_navigation_favorites) //只是一个占位图标
    .setContent(remoteViews) // 这里应用我们的视图
.setAutoCancel(true)
    .setContentIntent(pendingIntent)
    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

final Notification notification = notificationBuilder.build();

if (android.os.Build.VERSION.SDK_INT >= 16) {
    notification.bigContentView = remoteViews;
}

NotificationManager notificationManager =
    (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(uniqueIntentId, notification);

//别忘了在你的 build.gradle 文件中包含 picasso。
Picasso.with(context)
.load(avatar)
.into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId, notification);
```

然后在你的布局文件夹中定义一个布局：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:orientation="vertical">

    <ImageView
```

public or undirected social updates: The user has asked to be notified about them, but these notifications should never take precedence over urgent or direct communication.

PRIORITY_MIN -- Use for contextual or background information such as weather information or contextual location information. Minimum-priority notifications do not appear in the status bar. The user discovers them on expanding the notification shade.

References: [Material Design Guidelines - notifications](#)

Section 69.7: Set custom notification icon using `Picasso` library

```
PendingIntent pendingIntent = PendingIntent.getActivity(context,
    uniqueIntentId, intent, PendingIntent.FLAG_CANCEL_CURRENT);

final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.remote_view_notification);
remoteViews.setImageResource(R.id.remoteview_notification_icon,
R.mipmap.ic_navigation_favorites);

Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder =
    new NotificationCompat.Builder(context)
.setSmallIcon(R.mipmap.ic_navigation_favorites) //just dummy icon
    .setContent(remoteViews) // here we apply our view
    .setAutoCancel(true)
    .setContentIntent(pendingIntent)
    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

final Notification notification = notificationBuilder.build();

if (android.os.Build.VERSION.SDK_INT >= 16) {
    notification.bigContentView = remoteViews;
}

NotificationManager notificationManager =
    (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(uniqueIntentId, notification);

//don't forget to include picasso to your build.gradle file.
Picasso.with(context)
.load(avatar)
.into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId, notification);
```

And then define a layout inside your layouts folder:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:orientation="vertical">
```

<ImageView

```

    android:id="@+id/remoteview_notification_icon"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:layout_marginRight="2dp"
    android:layout_weight="0"
    android:scaleType="centerCrop"/>

```

第69.8节：调度通知

有时需要在特定时间显示通知，这在Android系统上并非易事，因为通知没有类似setTime()的方法。此示例概述了使用AlarmManager调度通知所需的步骤：

- 添加一个BroadcastReceiver，用于监听AndroidAlarmManager广播的Intent。

这是您根据Intent提供的额外信息构建通知的地方：

```

public class NotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 根据Intent构建通知
        Notification notification = new NotificationCompat.Builder(context)
            .setSmallIcon(R.drawable.ic_notification_small_icon)
            .setContentTitle(intent.getStringExtra("title", ""))
            .setContentText(intent.getStringExtra("text", ""))
            .build();
        // 显示通知
        NotificationManager manager = (NotificationManager)
            context.getSystemService(Context.NOTIFICATION_SERVICE);
        manager.notify(42, notification);
    }
}

```

- 在您的AndroidManifest.xml文件中注册BroadcastReceiver（否则接收器将无法接收任何来自AlarmManager的Intent）：

```

<receiver
    android:name=".NotificationReceiver"
    android:enabled="true" />

```

- 通过传递PendingIntent给你的BroadcastReceiver，并附带所需的Intent，来安排通知额外参数给系统的AlarmManager。你的BroadcastReceiver将在指定时间到达时接收该Intent并显示通知。以下方法用于安排通知：

```

public static void scheduleNotification(Context context, long time, String title, String
    text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    // 安排通知
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
    manager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, time, pending);
}

```

```

    android:id="@+id/remoteview_notification_icon"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:layout_marginRight="2dp"
    android:layout_weight="0"
    android:scaleType="centerCrop"/>

```

Section 69.8: Scheduling notifications

Sometimes it is required to display a notification at a specific time, a task that unfortunately is not trivial on the Android system, as there is no method setTime() or similar for notifications. This example outlines the steps needed to schedule notifications using the AlarmManager:

- Add a BroadcastReceiver that listens to Intents broadcasted by the Android AlarmManager.

This is the place where you build your notification based on the extras provided with the Intent:

```

public class NotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Build notification based on Intent
        Notification notification = new NotificationCompat.Builder(context)
            .setSmallIcon(R.drawable.ic_notification_small_icon)
            .setContentTitle(intent.getStringExtra("title", ""))
            .setContentText(intent.getStringExtra("text", ""))
            .build();
        // Show notification
        NotificationManager manager = (NotificationManager)
            context.getSystemService(Context.NOTIFICATION_SERVICE);
        manager.notify(42, notification);
    }
}

```

- Register the BroadcastReceiver in your AndroidManifest.xml file (otherwise the receiver won't receive any Intents from the AlarmManager):

```

<receiver
    android:name=".NotificationReceiver"
    android:enabled="true" />

```

- Schedule a notification by passing a PendingIntent for your BroadcastReceiver with the needed Intent extras to the system AlarmManager. Your BroadcastReceiver will receive the Intent once the given time has arrived and display the notification. The following method schedules a notification:

```

public static void scheduleNotification(Context context, long time, String title, String
    text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    // Schedule notification
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
    manager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, time, pending);
}

```

请注意，上述的42需要对每个计划通知唯一，否则PendingIntent将相互替换，导致不良后果！

4. 通过重建关联的PendingIntent并在系统上取消它来取消通知
AlarmManager。以下方法用于取消通知：

```
public static void cancelNotification(Context context, String title, String text) {  
    Intent intent = new Intent(context, NotificationReceiver.class);  
    intent.putExtra("title", title);  
    intent.putExtra("text", text);  
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,  
    PendingIntent.FLAG_UPDATE_CURRENT);  
    // 取消通知  
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);  
    manager.cancel(pending);  
}
```

请注意，上述的42需要与步骤3中的数字匹配！

Please note that the 42 above needs to be unique for each scheduled notification, otherwise the PendingIntents will replace each other causing undesired effects!

4. **Cancel a notification** by rebuilding the associated PendingIntent and canceling it on the system AlarmManager. The following method cancels a notification:

```
public static void cancelNotification(Context context, String title, String text) {  
    Intent intent = new Intent(context, NotificationReceiver.class);  
    intent.putExtra("title", title);  
    intent.putExtra("text", text);  
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,  
    PendingIntent.FLAG_UPDATE_CURRENT);  
    // Cancel notification  
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);  
    manager.cancel(pending);  
}
```

Note that the 42 above needs to match the number from step 3!

第70章：AlarmManager

第70.1节：如何取消闹钟

如果您想取消闹钟，但没有用于设置闹钟的原始PendingIntent的引用，您需要重新创建一个与最初创建时完全相同的PendingIntent。

AlarmManager认为两个Intent是相等的条件是：

如果它们的动作、数据、类型、类和类别相同。此比较不包括Intent中包含的任何额外数据。

通常，每个闹钟的请求代码定义为一个常量：

```
public static final int requestCode = 9999;
```

因此，对于如下简单的闹钟设置：

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, targetTimeInMillis, pendingIntent);
```

以下是如何创建一个新的 PendingIntent 引用，您可以使用它配合新的 AlarmManager 引用来取消闹钟：

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_NO_CREATE);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
if(pendingIntent != null) {
alarmManager.cancel(pendingIntent);
}
```

第70.2节：在所有Android版本上创建精确闹钟

随着越来越多的电池优化被引入Android系统，AlarmManager的方法也发生了显著变化（以允许更宽松的定时）。然而，对于某些应用来说，仍然需要在所有Android版本上尽可能精确。以下辅助方法使用所有平台上可用的最精确方法来调度一个PendingIntent：

```
public static void setExactAndAllowWhileIdle(AlarmManager alarmManager, int type, long
triggerAtMillis, PendingIntent operation) {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M){
        alarmManager.setExactAndAllowWhileIdle(type, triggerAtMillis, operation);
    } else if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
        alarmManager.setExact(type, triggerAtMillis, operation);
    } else {
        alarmManager.set(type, triggerAtMillis, operation);
    }
}
```

Chapter 70: AlarmManager

Section 70.1: How to Cancel an Alarm

If you want to cancel an alarm, and you don't have a reference to the original PendingIntent used to set the alarm, you need to recreate a PendingIntent exactly as it was when it was originally created.

An Intent is considered equal by the AlarmManager:

if their action, data, type, class, and categories are the same. This does not compare any extra data included in the intents.

Usually the request code for each alarm is defined as a constant:

```
public static final int requestCode = 9999;
```

So, for a simple alarm set up like this:

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, targetTimeInMillis, pendingIntent);
```

Here is how you would create a new PendingIntent reference that you can use to cancel the alarm with a new AlarmManager reference:

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_NO_CREATE);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
if(pendingIntent != null) {
    alarmManager.cancel(pendingIntent);
}
```

Section 70.2: Creating exact alarms on all Android versions

With more and more battery optimizations being put into the Android system over time, the methods of the AlarmManager have also significantly changed (to allow for more lenient timing). However, for some applications it is still required to be as exact as possible on all Android versions. The following helper uses the most accurate method available on all platforms to schedule a PendingIntent:

```
public static void setExactAndAllowWhileIdle(AlarmManager alarmManager, int type, long
triggerAtMillis, PendingIntent operation) {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M){
        alarmManager.setExactAndAllowWhileIdle(type, triggerAtMillis, operation);
    } else if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
        alarmManager.setExact(type, triggerAtMillis, operation);
    } else {
        alarmManager.set(type, triggerAtMillis, operation);
    }
}
```

}

第70.3节：API23及以上版本的Doze模式干扰AlarmManager

Android 6 (API23) 引入了Doze模式，该模式会干扰AlarmManager。它使用特定的维护窗口来处理闹钟，因此即使你使用了setExactAndAllowWhileIdle()，也无法确保闹钟在期望的时间点触发。

你可以通过手机的设置关闭该行为（设置/通用/电池&省电/电池使用/忽略优化或类似选项）

在你的应用内可以检查此设置……

```
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (pm.isIgnoringBatteryOptimizations(packageName)) {
    // 你的应用正在忽略Doze电池优化
}
```

……并最终显示相应的设置对话框：

```
Intent intent = new Intent();
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:" + packageName));
startActivity(intent);
```

第70.4节：稍后运行一个意图

1. 创建一个接收器。该类将接收意图并按您的需求处理它。

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理意图
        int requestCode = intent.getExtras().getInt("requestCode");
        ...
    }
}
```

2. 将意图交给AlarmManager。此示例将在1分钟内触发。

```
final int requestCode = 1337;
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
am.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 60000, pendingIntent);
```

}

Section 70.3: API23+ Doze mode interferes with AlarmManager

Android 6 (API23) introduced Doze mode which interferes with AlarmManager. It uses certain maintenance windows to handle alarms, so even if you used setExactAndAllowWhileIdle() you cannot make sure that your alarm fires at the desired point of time.

You can turn this behavior off for your app using your phone's settings (Settings/General/Battery & power saving/Battery usage/Ignore optimizations or similar)

Inside your app you can check this setting ...

```
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (pm.isIgnoringBatteryOptimizations(packageName)) {
    // your app is ignoring Doze battery optimization
}
```

... and eventually show the respective settings dialog:

```
Intent intent = new Intent();
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:" + packageName));
startActivity(intent);
```

Section 70.4: Run an intent at a later time

1. Create a receiver. This class will receive the intent and handle it how you wish.

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Handle intent
        int requestCode = intent.getExtras().getInt("requestCode");
        ...
    }
}
```

2. Give an intent to AlarmManager. This example will trigger the intent to be sent to AlarmReceiver after 1 minute.

```
final int requestCode = 1337;
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
am.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 60000, pendingIntent);
```

第71章：Handler

第71.1节：HandlerThread和线程间通信

由于Handler用于向线程的消息队列发送Message和Runnable，实现多线程间基于事件的通信变得简单。每个拥有Looper的线程都能够接收和处理消息。一个HandlerThread是实现了此类Looper的线程，例如主线程（UI线程）就实现了HandlerThread的功能。

为当前线程创建Handler

```
Handler handler = new Handler();
```

为主线程（UI线程）创建Handler

```
Handler handler = new Handler(Looper.getMainLooper());
```

从另一个线程向主线程发送Runnable

```
new Thread(new Runnable() {
    public void run() {
        // 这将在另一个线程上执行

        // 创建一个与主线程关联的Handler
        Handler handler = new Handler(Looper.getMainLooper());

        // 向主线程发布一个Runnable
        handler.post(new Runnable() {
            public void run() {
                // 这将在主线程上执行
            }
        });
    }
}).start();
```

为另一个HandlerThread创建Handler并向其发送事件

```
// 创建另一个线程
HandlerThread otherThread = new HandlerThread("name");

// 创建一个与另一个线程关联的Handler
Handler handler = new Handler(otherThread.getLooper());

// 向另一个线程发送一个事件
handler.post(new Runnable() {
    public void run() {
        // 这将在另一个线程上执行
    }
});
```

第71.2节：使用Handler创建一个定时器（类似于javax.swing.Timer）

如果你正在编写游戏或需要每隔几秒执行一段代码的程序，这会很有用。

```
import android.os.Handler;
```

Chapter 71: Handler

Section 71.1: HandlerThreads and communication between Threads

As Handlers are used to send Messages and [Runnables](#) to a Thread's message queue it's easy to implement event based communication between multiple Threads. Every Thread that has a Looper is able to receive and process messages. A HandlerThread is a Thread that implements such a Looper, for example the main Thread (UI Thread) implements the features of a HandlerThread.

Creating a Handler for the current Thread

```
Handler handler = new Handler();
```

Creating a Handler for the main Thread (UI Thread)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Send a Runnable from another Thread to the main Thread

```
new Thread(new Runnable() {
    public void run() {
        // this is executed on another Thread

        // create a Handler associated with the main Thread
        Handler handler = new Handler(Looper.getMainLooper());

        // post a Runnable to the main Thread
        handler.post(new Runnable() {
            public void run() {
                // this is executed on the main Thread
            }
        });
    }
}).start();
```

Creating a Handler for another HandlerThread and sending events to it

```
// create another Thread
HandlerThread otherThread = new HandlerThread("name");

// create a Handler associated with the other Thread
Handler handler = new Handler(otherThread.getLooper());

// post an event to the other Thread
handler.post(new Runnable() {
    public void run() {
        // this is executed on the other Thread
    }
});
```

Section 71.2: Use Handler to create a Timer (similar to javax.swing.Timer)

This can be useful if you're writing a game or something that needs to execute a piece of code every a few seconds.

```
import android.os.Handler;
```

```

public class Timer {
    private Handler handler;
    private boolean paused;

    private int interval;

    private Runnable task = new Runnable () {
        @Override
        public void run() {
            if (!paused) {
                runnable.run ();
                Timer.this.handler.postDelayed (this, interval);
            }
        }
    };

    private Runnable runnable;

    public int getInterval() {
        return interval;
    }

    public void setInterval(int interval) {
        this.interval = interval;
    }

    public void startTimer () {
        paused = false;
        handler.postDelayed (task, interval);
    }

    public void stopTimer () {
        paused = true;
    }

    public Timer (Runnable runnable, int interval, boolean started) {
        handler = new Handler ();
        this.runnable = runnable;
        this.interval = interval;
        if (started)
            startTimer ();
    }
}

```

示例用法：

```

Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)

```

这段代码将每秒打印一次“Hello”。

第71.3节：使用Handler在延迟一段时间后执行代码

1.5秒后执行代码：

```
Handler handler = new Handler();
```

```

public class Timer {
    private Handler handler;
    private boolean paused;

    private int interval;

    private Runnable task = new Runnable () {
        @Override
        public void run() {
            if (!paused) {
                runnable.run ();
                Timer.this.handler.postDelayed (this, interval);
            }
        }
    };

    private Runnable runnable;

    public int getInterval() {
        return interval;
    }

    public void setInterval(int interval) {
        this.interval = interval;
    }

    public void startTimer () {
        paused = false;
        handler.postDelayed (task, interval);
    }

    public void stopTimer () {
        paused = true;
    }

    public Timer (Runnable runnable, int interval, boolean started) {
        handler = new Handler ();
        this.runnable = runnable;
        this.interval = interval;
        if (started)
            startTimer ();
    }
}

```

Example usage:

```

Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)

```

This code will print "Hello" every second.

Section 71.3: Using a Handler to execute code after a delayed amount of time

Executing code after 1.5 seconds:

```
Handler handler = new Handler();
```

```

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        //你想在时间到后运行的代码
    }
}, 1500); //你想延迟的时间，单位为毫秒

```

每隔1秒重复执行代码：

```

Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        handler.postDelayed(this, 1000);
    }
}, 1000); //你想延迟的时间，单位为毫秒

```

第71.4节：停止Handler的执行

要停止Handler的执行，移除附加到它的回调，使用其内部运行的Runnable：

```

Runnable my_runnable = new Runnable() {
    @Override
    public void run() {
        // 你的代码写在这里
    }
};

public Handler handler = new Handler(); // 如果你想让这个handler控制UI中的某些内容，请使用 'new Handler(Looper.getMainLooper());'
// 启动handler
public void start() {
    handler.postDelayed(my_runnable, 10000);
}

// 停止handler
public void stop() {
    handler.removeCallbacks(my_runnable);
}

// 重置handler
public void restart() {
    handler.removeCallbacks(my_runnable);
    handler.postDelayed(my_runnable, 10000);
}

```

```

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        //The code you want to run after the time is up
    }
}, 1500); //the time you want to delay in milliseconds

```

Executing code repeatedly every 1 second:

```

Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        handler.postDelayed(this, 1000);
    }
}, 1000); //the time you want to delay in milliseconds

```

Section 71.4: Stop handler from execution

To stop the Handler from execution remove the callback attached to it using the runnable running inside it:

```

Runnable my_runnable = new Runnable() {
    @Override
    public void run() {
        // your code here
    }
};

public Handler handler = new Handler(); // use 'new Handler(Looper.getMainLooper());' if you want
// this handler to control something in the UI
// to start the handler
public void start() {
    handler.postDelayed(my_runnable, 10000);
}

// to stop the handler
public void stop() {
    handler.removeCallbacks(my_runnable);
}

// to reset the handler
public void restart() {
    handler.removeCallbacks(my_runnable);
    handler.postDelayed(my_runnable, 10000);
}

```

第72章：BroadcastReceiver

BroadcastReceiver（接收器）是Android组件，允许你注册系统或应用程序事件。一旦事件发生，所有注册该事件的接收器都会由Android运行时通知。

例如，广播通知屏幕已关闭、电池电量低或拍摄了照片。

应用程序也可以发起广播——例如，让其他应用知道某些数据已下载到设备并可供使用。

第72.1节：使用LocalBroadcastManager

*LocalBroadcastManager*用于在应用程序内部发送广播Intent，而不会暴露给不需要的监听器。

使用*LocalBroadcastManager*比直接使用*context.sendBroadcast()*更高效且更安全，因为你无需担心其他应用伪造的广播，这可能带来安全隐患。

下面是一个发送和接收本地广播的简单示例：

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("Some Action")) {
            //执行某些操作
        }
    }
});

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(mContext);
manager.registerReceiver(receiver, new IntentFilter("Some Action"));

// onReceive() 将会因这次调用而被调用：
manager.sendBroadcast(new Intent("Some Action")); // 另见 sendBroadcastSync

// 使用完毕后记得注销接收器：
manager.unregisterReceiver(receiver);
```

第72.2节：BroadcastReceiver基础

*BroadcastReceiver*用于接收由Android操作系统、其他应用或同一应用内发送的广播Intent。

每个Intent都通过一个Intent Filter创建，该过滤器需要一个字符串类型的action。Intent中还可以配置其他信息。

同样，*BroadcastReceiver*注册以接收具有特定Intent Filter的Intent。它们可以通过编程方式注册：

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 你的实现代码写在这里。
    }
}, new IntentFilter("Some Action"));
```

Chapter 72: BroadcastReceiver

BroadcastReceiver (receiver) is an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the Android runtime once this event happens.

for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.

Applications can also initiate broadcasts—for example, to let other applications know that some data has been downloaded to the device and is available for them to use.

Section 72.1: Using LocalBroadcastManager

LocalBroadcastManager is used to send Broadcast Intents within an application, without exposing them to unwanted listeners.

Using *LocalBroadcastManager* is more efficient and safer than using *context.sendBroadcast()* directly, because you don't need to worry about any broadcasts faked by other Applications, which may pose a security hazard.

Here is a simple example of sending and receiving local broadcasts:

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("Some Action")) {
            //Do something
        }
    }
};

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(mContext);
manager.registerReceiver(receiver, new IntentFilter("Some Action"));

// onReceive() will be called as a result of this call:
manager.sendBroadcast(new Intent("Some Action")); //See also sendBroadcastSync

//Remember to unregister the receiver when you are done with it:
manager.unregisterReceiver(receiver);
```

Section 72.2: BroadcastReceiver Basics

BroadcastReceivers are used to receive broadcast Intents that are sent by the Android OS, other apps, or within the same app.

Each Intent is created with an *Intent Filter*, which requires a String *action*. Additional information can be configured in the Intent.

Likewise, *BroadcastReceivers* register to receive Intents with a particular Intent Filter. They can be registered programmatically:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //Your implementation goes here.
    }
}, new IntentFilter("Some Action"));
```

或在AndroidManifest.xml文件中：

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

要接收该Intent，请将Action设置为Android操作系统、其他应用或API，或您自己应用中有文档说明的内容，使用 sendBroadcast：

```
mContext.sendBroadcast(new Intent("Some Action"));
```

此外，Intent可以包含信息，如字符串、基本类型和Parcelables，这些信息可以在 onReceive中查看。

第72.3节：广播接收器简介

广播接收器是一个安卓组件，允许你注册系统或应用程序事件。

接收器可以通过AndroidManifest.xml文件注册，或者通过Context的registerReceiver()

方法动态注册。

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 你的实现代码写在这里。
    }
}
```

这里我举了一个ACTION_BOOT_COMPLETED的例子，该动作由系统在Android完成启动过程后触发。

你可以在清单文件中这样注册接收器：

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED">
            </action>
        </intent-filter>
    </receiver>

```

设备启动后，onReceive()方法将被调用，然后你可以执行你的操作（例如启动服务、启动闹钟）。

第72.4节：使用有序广播

当您需要为广播接收器指定优先级时，使用有序广播。

在此示例中，firstReceiver 将始终比 secondReceiver 先接收广播：

or in the AndroidManifest.xml file:

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

To receive the Intent, set the Action to something documented by Android OS, by another app or API, or within your own application, using sendBroadcast:

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Additionally, the Intent can contain information, such as Strings, primitives, and Parcelables, that can be viewed in onReceive.

Section 72.3: Introduction to Broadcast receiver

A Broadcast receiver is an Android component which allows you to register for system or application events.

A receiver can be registered via the AndroidManifest.xml file or dynamically via the [Context.registerReceiver\(\)](#) method.

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //Your implementation goes here.
    }
}
```

Here I have taken an example of ACTION_BOOT_COMPLETED which is fired by the system once the Android has completed the boot process.

You can register a receiver in manifest file like this:

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED">
            </action>
        </intent-filter>
    </receiver>
</application>
```

Now device gets booted, onReceive() method will be called and then you can do your work (e.g. start a service, start an alarm).

Section 72.4: Using ordered broadcasts

Ordered broadcasts are used when you need to specify a priority for broadcast listeners.

In this example firstReceiver will receive broadcast always before than a secondReceiver:

```

final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// 为具有高优先级的第一个接收器设置意图过滤器
final IntentFilter firstFilter = new IntentFilter(action);
firstFilter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// 为具有低优先级的第二个接收器设置意图过滤器
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
final BroadcastReceiver secondReceiver = new MyReceiver();

// 注册我们的接收器
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// 发送有序广播
context.sendOrderedBroadcast(new Intent(action), null);

```

此外，广播接收器可以中止有序广播：

```

@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}

```

在这种情况下，所有优先级较低的接收器将不会接收到广播消息。

第72.5节：粘性广播

如果我们使用方法 `sendStickyBroadcast(intent)`，则对应的 `intent` 是粘性的，意味着你发送的 `intent` 在广播完成后仍然存在。顾名思义，粘性广播是一种机制，可以在广播完成后读取广播中的数据。这可以用于某种场景，比如你可能想在一个Activity的 `onCreate()`中检查 `intent` 中某个键的值，该值是在该 Activity 启动之前设置的。

```

Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);

```

第72.6节：以编程方式启用和禁用广播接收器

要启用或禁用一个BroadcastReceiver，我们需要获取PackageManager的引用，并且需要一个包含我们想要启用/禁用的接收器类的ComponentName对象：

```

ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();

```

现在我们可以调用以下方法来启用BroadcastReceiver：

```

packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);

```

```

final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// intent filter for first receiver with high priority
final IntentFilter firstFilter = new IntentFilter(action);
firstFilter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// intent filter for second receiver with low priority
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
final BroadcastReceiver secondReceiver = new MyReceiver();

// register our receivers
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// send ordered broadcast
context.sendOrderedBroadcast(new Intent(action), null);

```

Furthermore broadcast receiver can abort ordered broadcast:

```

@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}

```

in this case all receivers with lower priority will not receive a broadcast message.

Section 72.5: Sticky Broadcast

If we are using method `sendStickyBroadcast(intent)` the corresponding intent is sticky, meaning the intent you are sending stays around after broadcast is complete. A StickyBroadcast as the name suggests is a mechanism to read the data from a broadcast, after the broadcast is complete. This can be used in a scenario where you may want to check say in an Activity's `onCreate()` the value of a key in the intent before that Activity was launched.

```

Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);

```

Section 72.6: Enabling and disabling a Broadcast Receiver programmatically

To enable or disable a BroadcastReceiver, we need to get a reference to the PackageManager and we need a ComponentName object containing the class of the receiver we want to enable/disable:

```

ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();

```

Now we can call the following method to enable the BroadcastReceiver:

```

packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);

```

或者我们也可以使用COMPONENT_ENABLED_STATE_DISABLED来禁用接收器：

```
packageManager.setComponentEnabledSetting(  
    componentName,  
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,  
    PackageManager.DONT_KILL_APP);
```

第72.7节：LocalBroadcastManager示例

BroadcastReceiver基本上是一种通过操作系统传递Intent以执行特定操作的机制。一个经典的定义是

"BroadcastReceiver是一个Android组件，允许你注册系统或应用程序事件。"

LocalBroadcastManager是一种在应用进程内发送或接收广播的方式。这种机制有很多优点

- 1.由于数据保留在应用程序进程中，数据不会泄露。
- 2.本地广播(LocalBroadcasts)的解析速度更快，因为普通广播的解析是在操作系统运行时进行的。

LocalBroadcastManager 的一个简单示例是：

发送者活动 (SenderActivity)

```
Intent intent = new Intent("anEvent");  
intent.putExtra("key", "这是一个事件");  
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

接收者活动 (ReceiverActivity)

1. 注册接收器

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,  
    new IntentFilter("anEvent"));
```

2. 当接收器被调用时执行操作的具体对象

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // 在这里执行操作。  
    }  
};
```

3. 当视图不再可见时取消注册。

```
@Override  
protected void onPause() {  
    // 由于活动即将关闭，取消注册。  
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);  
    super.onDestroy();  
}
```

Or we can instead use COMPONENT_ENABLED_STATE_DISABLED to disable the receiver:

```
packageManager.setComponentEnabledSetting(  
    componentName,  
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,  
    PackageManager.DONT_KILL_APP);
```

Section 72.7: Example of a LocalBroadcastManager

A BroadcastReceiver is basically a mechanism to relay Intents through the OS to perform specific actions. A classic definition being

"A Broadcast receiver is an Android component which allows you to register for system or application events."

LocalBroadcastManager is a way to send or receive broadcasts within an application process. This mechanism has a lot of advantages

1. since the data remains inside the application process, the data cannot be leaked.
2. LocalBroadcasts are resolved faster, since the resolution of a normal broadcast happens at runtime throughout the OS.

A simple example of a LocalBroadcastManager is:

SenderIdActivity

```
Intent intent = new Intent("anEvent");  
intent.putExtra("key", "This is an event");  
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Register a receiver

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,  
    new IntentFilter("anEvent"));
```

2. A concrete object for performing action when the receiver is called

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // perform action here.  
    }  
};
```

3. unregister when the view is not visible any longer.

```
@Override  
protected void onPause() {  
    // Unregister since the activity is about to be closed.  
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);  
    super.onDestroy();  
}
```

第72.8节：Android 停止状态

从 Android 3.1 开始，所有应用安装后都会处于停止状态。在停止状态下，应用不会因任何原因运行，除非手动启动一个活动，或发送一个明确的意图 (explicit intent)，该意图针对活动、服务或广播。

编写直接安装 APK 的系统应用时，请注意新安装的应用在移出停止状态之前不会接收任何广播。

激活应用的一个简单方法是向该应用发送一个明确的广播，因为大多数应用都会实现此功能。`INSTALL_REFERRER`，我们可以将其用作钩子点

扫描已安装应用的清单，并向每个接收器发送显式广播：

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

第72.9节：通过自定义广播接收器通信两个活动

你可以让两个活动进行通信，使活动A能够接收到活动B中发生事件的通知。

活动A

```
final String eventName = "your.package.goes.here.EVENT";

@Override
protected void onCreate(Bundle savedInstanceState) {
    registerEventReceiver();
    super.onCreate(savedInstanceState);
}

@Override
protected void onDestroy() {
    unregisterEventReceiver(eventReceiver);
    super.onDestroy();
}

private void registerEventReceiver() {
    IntentFilter eventFilter = new IntentFilter();
    eventFilter.addAction(eventName);
    registerReceiver(eventReceiver, eventFilter);
}

private BroadcastReceiver eventReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //当活动B中的广播被触发时，这段代码将被执行
    }
};
```

活动B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
```

Section 72.8: Android stopped state

Starting with Android 3.1 all applications, upon installation, are placed in a stopped state. While in stopped state, the application will not run for any reason, except by a manual launch of an activity, or an **explicit** intent that addresses an activity, service or broadcast.

When writing system app that installs APKs directly, please take into account that the newly installed APP won't receive any broadcasts until moved into a non stopped state.

An easy way to activate an app is to send an explicit broadcast to this app. as most apps implement `INSTALL_REFERRER`, we can use it as a hooking point

Scan the manifest of the installed app, and send an explicit broadcast to each receiver:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Section 72.9: Communicate two activities through custom Broadcast receiver

You can communicate two activities so that Activity A can be notified of an event happening in Activity B.

Activity A

```
final String eventName = "your.package.goes.here.EVENT";

@Override
protected void onCreate(Bundle savedInstanceState) {
    registerEventReceiver();
    super.onCreate(savedInstanceState);
}

@Override
protected void onDestroy() {
    unregisterEventReceiver(eventReceiver);
    super.onDestroy();
}

private void registerEventReceiver() {
    IntentFilter eventFilter = new IntentFilter();
    eventFilter.addAction(eventName);
    registerReceiver(eventReceiver, eventFilter);
}

private BroadcastReceiver eventReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //This code will be executed when the broadcast in activity B is launched
    }
};
```

Activity B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
```

```

Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}

```

当然，你可以通过向Intent中添加额外信息来给广播传递更多数据，这些Intent会在活动之间传递。为了保持示例尽可能简单，这里未添加额外内容。

第72.10节：用于处理 BOOT_COMPLETED事件的BroadcastReceiver

下面的示例展示了如何创建一个BroadcastReceiver，该接收器能够接收BOOT_COMPLETED事件。这样，设备一开机，你就可以启动一个Service或启动一个Activity。

此外，您可以使用BOOT_COMPLETED事件来恢复您的闹钟，因为设备关闭电源时闹钟会被销毁。

注意：用户需要至少启动过一次应用程序，您才能接收到BOOT_COMPLETED动作。

AndroidManifest.xml

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.example" >
    ...
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    ...
    <application>
        ...
        <receiver android:name="com.test.example.MyCustomBroadcastReceiver">
            <intent-filter>
                <!-- 注册以接收 BOOT_COMPLETED 事件 -->
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>

```

MyCustomBroadcastReceiver.java

```

public class MyCustomBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(action != null) {
            if (action.equals(Intent.ACTION_BOOT_COMPLETED)) {
                // 待办事项：处理启动完成事件的代码
                // 待办事项：我可以启动服务.. 显示通知... 启动活动
            }
        }
    }
}

```

```

Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}

```

Of course you can add more information to the broadcast adding extras to the Intent that is passed between the activities. Not added to keep the example as simple as possible.

Section 72.10: BroadcastReceiver to handle BOOT_COMPLETED events

Example below shows how to create a BroadcastReceiver which is able to receive BOOT_COMPLETED events. This way, you are able to start a Service or start an Activity as soon device was powered up.

Also, you can use BOOT_COMPLETED events to restore your alarms since they are destroyed when device is powered off.

NOTE: The user needs to have started the application at least once before you can receive the BOOT_COMPLETED action.

AndroidManifest.xml

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.example" >
    ...
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    ...
    <application>
        ...
        <receiver android:name="com.test.example.MyCustomBroadcastReceiver">
            <intent-filter>
                <!-- REGISTER TO RECEIVE BOOT_COMPLETED EVENTS -->
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>

```

MyCustomBroadcastReceiver.java

```

public class MyCustomBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(action != null) {
            if (action.equals(Intent.ACTION_BOOT_COMPLETED)) {
                // TO-DO: Code to handle BOOT_COMPLETED EVENT
                // TO-DO: I can start an service.. display a notification... start an activity
            }
        }
    }
}

```

第72.11节：蓝牙广播接收器

在你的清单文件中添加权限

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

在你的Fragment（或Activity）中

- 添加接收器方法

```
private BroadcastReceiver mBluetoothStatusChangedReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final Bundle extras = intent.getExtras();
        final int bluetoothState = extras.getInt(Constants.BUNDLE_BLUETOOTH_STATE);
        switch(bluetoothState) {
            case BluetoothAdapter.STATE_OFF:
                // 蓝牙关闭
                break;
            case BluetoothAdapter.STATE_TURNING_OFF:
                // 正在关闭
                break;
            case BluetoothAdapter.STATE_ON:
                // 蓝牙开启
                break;
            case BluetoothAdapter.STATE_TURNING_ON:
                // 正在开启
                break;
        }
    }
};
```

注册广播

- 在 onResume() 中调用此方法

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

注销广播

- 在 onPause() 中调用此方法

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // 用于 Beacon 功能
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Section 72.11: Bluetooth Broadcast receiver

add permission in your manifest file

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

In your Fragment(or Activity)

- Add the receiver method

```
private BroadcastReceiver mBluetoothStatusChangedReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final Bundle extras = intent.getExtras();
        final int bluetoothState = extras.getInt(Constants.BUNDLE_BLUETOOTH_STATE);
        switch(bluetoothState) {
            case BluetoothAdapter.STATE_OFF:
                // Bluetooth OFF
                break;
            case BluetoothAdapter.STATE_TURNING_OFF:
                // Turning OFF
                break;
            case BluetoothAdapter.STATE_ON:
                // Bluetooth ON
                break;
            case BluetoothAdapter.STATE_TURNING_ON:
                // Turning ON
                break;
        }
    }
};
```

Register broadcast

- Call this method on onResume()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Unregister broadcast

- Call this method on onPause()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon 機能用
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

第73章：UI 生命周期

第73.1节：内存修剪时保存数据

```
public class ExampleActivity extends Activity {  
  
    private final static String EXAMPLE_ARG = "example_arg";  
    private int mArg;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_example);  
  
        if(savedInstanceState != null) {  
            mArg = savedInstanceState.getInt(EXAMPLE_ARG);  
        }  
    }  
  
    @Override  
    public void onSaveInstanceState(Bundle outState) {  
        super.onSaveInstanceState(outState);  
        outState.putInt(EXAMPLE_ARG, mArg);  
    }  
}
```

说明

那么，这里发生了什么？

Android系统总是尽力释放尽可能多的内存。因此，如果你的活动处于后台，而另一个前台活动需要分配内存，Android系统会调用你活动的 `onTrimMemory()` 方法。

但这并不意味着你的所有属性都应该消失。你应该做的是将它们保存到一个 `Bundle` 对象中。`Bundle` 对象在内存管理方面更为高效。在 `Bundle` 中，每个对象都由唯一的文本序列标识——在上面的例子中，整数值变量 `mArg` 是通过引用名 `EXAMPLE_ARG` 保存的。当活动被重新创建时，应从 `Bundle` 对象中提取旧值，而不是从头重新创建它们。

Chapter 73: UI Lifecycle

Section 73.1: Saving data on memory trimming

```
public class ExampleActivity extends Activity {  
  
    private final static String EXAMPLE_ARG = "example_arg";  
    private int mArg;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_example);  
  
        if(savedInstanceState != null) {  
            mArg = savedInstanceState.getInt(EXAMPLE_ARG);  
        }  
    }  
  
    @Override  
    public void onSaveInstanceState(Bundle outState) {  
        super.onSaveInstanceState(outState);  
        outState.putInt(EXAMPLE_ARG, mArg);  
    }  
}
```

Explanation

So, what is happening here?

The Android system will always strive to clear as much memory as it can. So, if your activity is down to the background, and another foreground activity is demanding its share, the Android system will call `onTrimMemory()` on your activity.

But that doesn't mean that all your properties should vanish. What you should do is to save them into a `Bundle` object. `Bundle` objects are much better handled memory wise. Inside a bundle every object is identified by unique text sequence - in the example above integer value variable `mArg` is held under reference name `EXAMPLE_ARG`. And when the activity is recreated extract your old values from the `Bundle` object instead of recreating them from scratch

第74章：HttpURLConnection

第74.1节：创建HttpURLConnection

为了创建一个新的Android HTTP客户端HttpURLConnection，调用URL实例上的openConnection()。由于openConnection()返回一个URLConnection，你需要显式地将返回值进行类型转换。

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// 对连接进行操作
```

如果你正在创建一个新的URL，也必须处理与URL解析相关的异常。

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // 对连接进行操作
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

一旦响应体被读取且不再需要连接，应通过调用disconnect()关闭连接。

下面是一个示例：

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // 对连接进行操作
} 最后 {
    连接。断开连接();
}
```

第74.2节：发送HTTP GET请求

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // 读取输入流
    // 在此示例中，我仅读取流的第一行
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} 最后 {
    连接。断开连接();
}
```

请注意，上述示例中未处理异常。一个完整的示例，包括（简单的）异常处理，将是：

```
URL url;
HttpURLConnection connection = null;
```

Chapter 74: HttpURLConnection

Section 74.1: Creating an HttpURLConnection

In order to create a new Android HTTP Client [HttpURLConnection](#), call `openConnection()` on a URL instance. Since `openConnection()` returns a [URLConnection](#), you need to explicitly cast the returned value.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

If you are creating a new [URL](#), you also have to handle the exceptions associated with URL parsing.

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Once the response body has been read and the connection is no longer required, the connection should be closed by calling `disconnect()`.

Here is an example:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Section 74.2: Sending an HTTP GET request

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} finally {
    connection.disconnect();
}
```

Please note that exceptions are not handled in the example above. A full example, including (a trivial) exception handling, would be:

```
URL url;
HttpURLConnection connection = null;
```

```

try {
    url = new URL("http://example.com");
    connection = (HttpURLConnection) url.openConnection();
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // 读取输入流
    // 在此示例中，我仅读取流的第一行
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}

```

第74.3节：读取HTTP GET请求的主体

```

URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // 使用字符串构建器缓存响应主体
    // 从输入流读取。
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append("\n");
    }

    // 直接使用字符串构建器,
    // 或将其转换为字符串
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    // 连接。断开连接();
}

```

请注意，上述示例中未处理异常。

第74.4节：发送带有参数的HTTP POST请求

使用HashMap存储应通过POST参数发送到服务器的参数：

```
HashMap<String, String> params;
```

一旦填充了params HashMap，创建将用于发送它们到服务器的StringBuilder：

```

StringBuilder sbParams = new StringBuilder();
int i = 0;
for (String key : params.keySet()) {
    try {

```

```

try {
    url = new URL("http://example.com");
    connection = (HttpURLConnection) url.openConnection();
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}

```

Section 74.3: Reading the body of an HTTP GET request

```

URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // use a string builder to bufferize the response body
    // read from the input strea.
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append('\n');
    }

    // use the string builder directly,
    // or convert it into a String
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    connection.disconnect();
}

```

Please note that exceptions are not handled in the example above.

Section 74.4: Sending an HTTP POST request with parameters

Use a HashMap to store the parameters that should be sent to the server through POST parameters:

```
HashMap<String, String> params;
```

Once the params HashMap is populated, create the StringBuilder that will be used to send them to the server:

```

StringBuilder sbParams = new StringBuilder();
int i = 0;
for (String key : params.keySet()) {
    try {

```

```

        if (i != 0){
            sbParams.append("&");
        }
        sbParams.append(key).append("=");
        .append(URLEncoder.encode(params.get(key), "UTF-8"));

    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    i++;
}

```

然后，创建 HttpURLConnection，打开连接，并发送 POST 参数：

```

try{
    String url = "http://www.example.com/test.php";
    URL urlObj = new URL(url);
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Accept-Charset", "UTF-8");

    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);

    conn.connect();

    String paramsString = sbParams.toString();

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());
    wr.writeBytes(paramsString);
    wr.flush();
    wr.close();
} catch (IOException e) {
    e.printStackTrace();
}

```

然后接收服务器返回的结果：

```

try {
    InputStream in = new BufferedInputStream(conn.getInputStream());
    BufferedReader reader = new BufferedReader(new InputStreamReader(in));
    StringBuilder result = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        result.append(line);
    }

    Log.d("test", "result from server: " + result.toString());

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (conn != null) {
        conn.disconnect();
    }
}

```

```

if (i != 0){
    sbParams.append("&");
}
sbParams.append(key).append("=");
.append(URLEncoder.encode(params.get(key), "UTF-8"));

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
i++;
}

```

Then, create the HttpURLConnection, open the connection, and send the POST parameters:

```

try{
    String url = "http://www.example.com/test.php";
    URL urlObj = new URL(url);
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Accept-Charset", "UTF-8");

    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);

    conn.connect();

    String paramsString = sbParams.toString();

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());
    wr.writeBytes(paramsString);
    wr.flush();
    wr.close();
} catch (IOException e) {
    e.printStackTrace();
}

```

Then receive the result that the server sends back:

```

try {
    InputStream in = new BufferedInputStream(conn.getInputStream());
    BufferedReader reader = new BufferedReader(new InputStreamReader(in));
    StringBuilder result = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        result.append(line);
    }

    Log.d("test", "result from server: " + result.toString());

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (conn != null) {
        conn.disconnect();
    }
}

```

第74.5节：一个多用途的HttpURLConnection类，用于处理所有类型的HTTP请求

以下类可以作为一个单一类来处理GET、POST、PUT、PATCH及其他请求：

```
class APIResponseObject{
    int responseCode;
    String response;

APIResponseObject(int responseCode, String response)
{
    this.responseCode = responseCode;
    this.response = response;
}

public class APIAccessTask extends AsyncTask<String,Void,APIResponseObject> {
    URL requestUrl;
    上下文 context;
    HttpURLConnection urlConnection;
    List<Pair<String, String>> postData, headerData;
    String method;
    int responseCode = HttpURLConnection.HTTP_OK;

    interface OnCompleteListener{
        void onComplete(APIResponseObject result);
    }

    public OnCompleteListener delegate = null;

    APIAccessTask(Context context, String requestUrl, String method, OnCompleteListener delegate){
        this.context = context;
        this.delegate = delegate;
        this.method = method;
        try {
            this.requestUrl = new URL(requestUrl);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String, String>> postData, OnCompleteListener delegate){
        this(context, requestUrl, method, delegate);
        this.postData = postData;
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String, String>> postData,
List<Pair<String, String>> headerData, OnCompleteListener delegate ){
        this(context, requestUrl, method, postData, delegate);
        this.headerData = headerData;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
}
```

Section 74.5: A multi-purpose HttpURLConnection class to handle all types of HTTP requests

The following class can be used as a single class that can handle GET, POST, PUT, PATCH, and other requests:

```
class APIResponseObject{
    int responseCode;
    String response;

APIResponseObject(int responseCode, String response)
{
    this.responseCode = responseCode;
    this.response = response;
}

public class APIAccessTask extends AsyncTask<String,Void,APIResponseObject> {
    URL requestUrl;
    Context context;
    HttpURLConnection urlConnection;
    List<Pair<String, String>> postData, headerData;
    String method;
    int responseCode = HttpURLConnection.HTTP_OK;

    interface OnCompleteListener{
        void onComplete(APIResponseObject result);
    }

    public OnCompleteListener delegate = null;

    APIAccessTask(Context context, String requestUrl, String method, OnCompleteListener delegate){
        this.context = context;
        this.delegate = delegate;
        this.method = method;
        try {
            this.requestUrl = new URL(requestUrl);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String, String>> postData, OnCompleteListener delegate){
        this(context, requestUrl, method, delegate);
        this.postData = postData;
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String, String>> postData,
List<Pair<String, String>> headerData, OnCompleteListener delegate ){
        this(context, requestUrl, method, postData, delegate);
        this.headerData = headerData;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
}
```

```

protected APIResponseObject doInBackground(String... params) {
    Log.d("debug", "url = "+ requestUrl);
    try {
        urlConnection = (HttpURLConnection) requestUrl.openConnection();

        if(headerData != null) {
            for (Pair pair : headerData) {
                urlConnection.setRequestProperty(pair.first.toString(),pair.second.toString());
            }
        }

        urlConnection.setDoInput(true);
        urlConnection.setChunkedStreamingMode(0);
        urlConnection.setRequestMethod(method);
        urlConnection.connect();

        StringBuilder sb = new StringBuilder();

        if(!(method.equals("GET"))) {
            OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
            writer.write(getpostDataString(postData));
            writer.flush();
            writer.close();
            out.close();
        }

        urlConnection.connect();
        responseCode = urlConnection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            BufferedReader reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
            String line;

            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
        }

        return new APIResponseObject(responseCode, sb.toString());
    }
    catch(Exception ex){
        ex.printStackTrace();
    }
    return null;
}

@Override
protected void onPostExecute(APIResponseObject result) {
    delegate.onComplete(result);
    super.onPostExecute(result);
}

private String getpostDataString(List<Pair<String, String>> params) throws
UnsupportedEncodingException {
    StringBuilder result = new StringBuilder();
    boolean first = true;
    for(Pair<String, String> pair : params){
        if (first)
            first = false;
        else
            result.append("&");
        result.append(pair.first);
        result.append("=");
        result.append(pair.second);
    }
}

```

```

protected APIResponseObject doInBackground(String... params) {
    Log.d("debug", "url = "+ requestUrl);
    try {
        urlConnection = (HttpURLConnection) requestUrl.openConnection();

        if(headerData != null) {
            for (Pair pair : headerData) {
                urlConnection.setRequestProperty(pair.first.toString(),pair.second.toString());
            }
        }

        urlConnection.setDoInput(true);
        urlConnection.setChunkedStreamingMode(0);
        urlConnection.setRequestMethod(method);
        urlConnection.connect();

        StringBuilder sb = new StringBuilder();

        if(!(method.equals("GET"))) {
            OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
            writer.write(getpostDataString(postData));
            writer.flush();
            writer.close();
            out.close();
        }

        urlConnection.connect();
        responseCode = urlConnection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            BufferedReader reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
            String line;

            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
        }

        return new APIResponseObject(responseCode, sb.toString());
    }
    catch(Exception ex){
        ex.printStackTrace();
    }
    return null;
}

@Override
protected void onPostExecute(APIResponseObject result) {
    delegate.onComplete(result);
    super.onPostExecute(result);
}

private String getpostDataString(List<Pair<String, String>> params) throws
UnsupportedEncodingException {
    StringBuilder result = new StringBuilder();
    boolean first = true;
    for(Pair<String, String> pair : params){
        if (first)
            first = false;
        else
            result.append("&");
        result.append(pair.first);
        result.append("=");
        result.append(pair.second);
    }
}

```

```

result.append(URLEncoder.encode(pair.first, "UTF-8"));
    result.append("=");
result.append(URLEncoder.encode(pair.second, "UTF-8"));
}
return result.toString();
}

```

用法

根据是否需要发送POST数据或任何额外的

头部，使用该类提供的任意构造函数。

onComplete()方法将在数据获取完成时被调用。数据作为APIResponseObject类的对象返回，该对象包含表示请求HTTP状态码的状态码和包含响应内容的字符串。你可以在你的类中解析该响应，例如XML或JSON格式。

调用该类对象的execute()方法以执行请求，如以下示例所示：

```

class MainClass {
    String url = "https://example.com./api/v1/ex";
    String method = "POST";
    List<Pair<String, String>> postData = new ArrayList<>();

    postData.add(new Pair<>("email", "whatever"));
    postData.add(new Pair<>("password", "whatever"));

    new APIAccessTask(MainActivity.this, url, method, postData,
        new APIAccessTask.OnCompleteListener() {
            @Override
            public void onComplete(APIResponseObject result) {
                if (result.responseCode == HttpURLConnection.HTTP_OK) {
                    String str = result.response;
                    // 在这里进行您的 XML/JSON 解析
                }
            }
        }).execute();
}

```

第74.6节：使用 HttpURLConnection 处理 multipart/form-data

创建自定义类以调用 multipart/form-data HttpURLConnection 请求

MultipartUtility.java

```

public class MultipartUtility {private fi
    nal String boundary;private static fi
    nal String LINE_FEED = "\r\n";private HttpURLConnection h
    ttpConn;
    private String charset;
    private OutputStream outputStream;
    private PrintWriter writer;

    /**
     * 此构造函数初始化一个新的 HTTP POST 请求，内容类型
     * 设置为 multipart/form-data
     *
     * @param requestURL
     * @param charset
     */

```

```

result.append(URLEncoder.encode(pair.first, "UTF-8"));
    result.append("=");
result.append(URLEncoder.encode(pair.second, "UTF-8"));
}
return result.toString();
}

```

Usage

Use any of the given constructors of the class depending on whether you need to send POST data or any extra headers.

The onComplete() method will be called when the data fetching is complete. The data is returned as an object of the APIResponseObject class, which has a status code stating the HTTP status code of the request and a string containing the response. You can parse this response in your class, i.e. XML or JSON.

Call execute() on the object of the class to execute the request, as shown in the following example:

```

class MainClass {
    String url = "https://example.com./api/v1/ex";
    String method = "POST";
    List<Pair<String, String>> postData = new ArrayList<>();

    postData.add(new Pair<>("email", "whatever"));
    postData.add(new Pair<>("password", "whatever"));

    new APIAccessTask(MainActivity.this, url, method, postData,
        new APIAccessTask.OnCompleteListener() {
            @Override
            public void onComplete(APIResponseObject result) {
                if (result.responseCode == HttpURLConnection.HTTP_OK) {
                    String str = result.response;
                    // Do your XML/JSON parsing here
                }
            }
        }).execute();
}

```

Section 74.6: Use HttpURLConnection for multipart/form-data

Create custom class for calling multipart/form-data HttpURLConnection request

MultipartUtility.java

```

public class MultipartUtility {
    private final String boundary;
    private static final String LINE_FEED = "\r\n";
    private HttpURLConnection httpConn;
    private String charset;
    private OutputStream outputStream;
    private PrintWriter writer;

    /**
     * This constructor initializes a new HTTP POST request with content type
     * is set to multipart/form-data
     *
     * @param requestURL
     * @param charset
     */

```

```

* @throws IOException
*/
public MultipartUtility(String requestURL, String charset)
    throws IOException {
    this.charset = charset;

    // 基于时间戳创建唯一边界
    boundary = "===" + System.currentTimeMillis() + "==";
    URL url = new URL(requestURL);
    httpConn = (HttpURLConnection) url.openConnection();
    httpConn.setUseCaches(false);
    httpConn.setDoOutput(true); // 表示POST方法
    httpConn.setDoInput(true);
    httpConn.setRequestProperty("Content-Type",
        "multipart/form-data; boundary=" + boundary);
    outputStream = httpConn.getOutputStream();
    writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
        true);
}

/**
* 向请求中添加一个表单字段
*
* @param name 字段名
* @param value 字段值
*/
public void addFormField(String name, String value) {
    writer.append("--" + boundary).append(LINE_FEED);
    writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
        .append(LINE_FEED);
    writer.append("Content-Type: text/plain; charset=" + charset).append(
        LINE_FEED);
    writer.append(LINE_FEED);
    writer.append(value).append(LINE_FEED);
    writer.flush();
}

/**
* 向请求中添加一个上传文件部分
*
* @param fieldName <input type="file" name="..." /> 中的 name 属性
* @param uploadFile 需要上传的文件
* @throws IOException
*/
public void addFilePart(String fieldName, File uploadFile)
    throws IOException {
    String fileName = uploadFile.getName();
    writer.append("--" + boundary).append(LINE_FEED);
    writer.append(
        "Content-Disposition: form-data; name=\"" + fieldName
            + "\"; filename=\"" + fileName + "\"")
        .append(LINE_FEED);
    writer.append(
        "Content-Type: "
            + URLConnection.guessContentTypeFromName(fileName))
        .append(LINE_FEED);
    writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
    writer.append(LINE_FEED);
    writer.flush();

    FileInputStream inputStream = new FileInputStream(uploadFile);
    byte[] buffer = new byte[4096];

```

```

* @throws IOException
*/
public MultipartUtility(String requestURL, String charset)
    throws IOException {
    this.charset = charset;

    // creates a unique boundary based on time stamp
    boundary = "===" + System.currentTimeMillis() + "===";
    URL url = new URL(requestURL);
    httpConn = (HttpURLConnection) url.openConnection();
    httpConn.setUseCaches(false);
    httpConn.setDoOutput(true); // indicates POST method
    httpConn.setDoInput(true);
    httpConn.setRequestProperty("Content-Type",
        "multipart/form-data; boundary=" + boundary);
    outputStream = httpConn.getOutputStream();
    writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
        true);
}

/**
* Adds a form field to the request
*
* @param name field name
* @param value field value
*/
public void addFormField(String name, String value) {
    writer.append("--" + boundary).append(LINE_FEED);
    writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
        .append(LINE_FEED);
    writer.append("Content-Type: text/plain; charset=" + charset).append(
        LINE_FEED);
    writer.append(LINE_FEED);
    writer.append(value).append(LINE_FEED);
    writer.flush();
}

/**
* Adds a upload file section to the request
*
* @param fieldName name attribute in <input type="file" name="..." />
* @param uploadFile a File to be uploaded
* @throws IOException
*/
public void addFilePart(String fieldName, File uploadFile)
    throws IOException {
    String fileName = uploadFile.getName();
    writer.append("--" + boundary).append(LINE_FEED);
    writer.append(
        "Content-Disposition: form-data; name=\"" + fieldName
            + "\"; filename=\"" + fileName + "\"")
        .append(LINE_FEED);
    writer.append(
        "Content-Type: "
            + URLConnection.guessContentTypeFromName(fileName))
        .append(LINE_FEED);
    writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
    writer.append(LINE_FEED);
    writer.flush();

    FileInputStream inputStream = new FileInputStream(uploadFile);
    byte[] buffer = new byte[4096];

```

```

int bytesRead = -1;
while ((bytesRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, bytesRead);
}
outputStream.flush();
inputStream.close();
writer.append(LINE_FEED);
writer.flush();
}

/**
* 向请求添加一个头字段。
*/
* @param name - 头字段的名称
* @param value - 头字段的值
*/
public void addHeaderField(String name, String value) {
    writer.append(name + ":" + value).append(LINE_FEED);
    writer.flush();
}

/**
* 完成请求并接收来自服务器的响应。
*/
* @return 如果服务器返回状态为OK，则返回字符串列表作为响应，否则抛出异常。
* @throws IOException
*/
public List<String> finish() throws IOException {
    List<String> response = new ArrayList<String>();
    writer.append(LINE_FEED).flush();
    writer.append(" -- " + boundary + " -- ").append(LINE_FEED);
    writer.close();

    // 先检查服务器的状态码
    int status = httpConn.getResponseCode();
    if (status == HttpURLConnection.HTTP_OK) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
            httpConn.getInputStream()));
        String line = null;
        while ((line = reader.readLine()) != null) {
            response.add(line);
        }
    reader.close();
        httpConn.disconnect();
    } else {
        throw new IOException("服务器返回非OK状态: " + status);
    }
    return response;
}
}

```

以异步方式使用它

```

MultipartUtility multipart = new MultipartUtility(requestURL, charset);

// 在你的情况下你没有添加表单数据，所以忽略这部分
/*这是用来添加参数值的*/
for (int i = 0; i < myFormDataArray.size(); i++) {
    multipart.addFormField(myFormDataArray.get(i).getParamName(),
        myFormDataArray.get(i).getParamValue());
}

```

```

int bytesRead = -1;
while ((bytesRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, bytesRead);
}
outputStream.flush();
inputStream.close();
writer.append(LINE_FEED);
writer.flush();
}

/**
* Adds a header field to the request.
*/
* @param name - name of the header field
* @param value - value of the header field
*/
public void addHeaderField(String name, String value) {
    writer.append(name + ":" + value).append(LINE_FEED);
    writer.flush();
}

/**
* Completes the request and receives response from the server.
*/
* @return a list of Strings as response in case the server returned
* status OK, otherwise an exception is thrown.
* @throws IOException
*/
public List<String> finish() throws IOException {
    List<String> response = new ArrayList<String>();
    writer.append(LINE_FEED).flush();
    writer.append(" -- " + boundary + " -- ").append(LINE_FEED);
    writer.close();

    // checks server's status code first
    int status = httpConn.getResponseCode();
    if (status == HttpURLConnection.HTTP_OK) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
            httpConn.getInputStream()));
        String line = null;
        while ((line = reader.readLine()) != null) {
            response.add(line);
        }
        reader.close();
        httpConn.disconnect();
    } else {
        throw new IOException("Server returned non-OK status: " + status);
    }
    return response;
}
}

```

Use it (Async way)

```

MultipartUtility multipart = new MultipartUtility(requestURL, charset);

// In your case you are not adding form data so ignore this
/*This is to add parameter values */
for (int i = 0; i < myFormDataArray.size(); i++) {
    multipart.addFormField(myFormDataArray.get(i).getParamName(),
        myFormDataArray.get(i).getParamValue());
}

```

```

    }

// 在这里添加你的文件。
/*这是用来添加文件内容的*/
for (int i = 0; i < myFileArray.size(); i++) {
    multipart.addFilePart(myFileArray.getParamName(),
        new File(myFileArray.getFileName()));
}

List<String> response = multipart.finish();
Debug.e(TAG, "SERVER REPLIED:");
for (String line : response) {
    Debug.e(TAG, "上传文件响应:::" + line);
}
// 在这里获取你的服务器响应。
responseString = line;
}

```

第74.7节：使用HttpURLConnection上传（POST）文件

经常需要将文件发送/上传到远程服务器，例如图像、视频、音频或应用数据库的备份到远程私有服务器。假设服务器期望接收带有内容的POST请求，下面是一个在Android中完成此任务的简单示例。

文件上传是通过multipart/form-data POST请求发送的。实现起来非常简单：

```

URL url = new URL(postTarget);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

String auth = "Bearer " + oauthToken;
connection.setRequestProperty("Authorization", basicAuth);

String boundary = UUID.randomUUID().toString();
connection.setRequestMethod("POST");
connection.setDoOutput(true);
connection.setRequestProperty("Content-Type", "multipart/form-data;boundary=" + boundary);

DataOutputStream request = new DataOutputStream(uc.getOutputStream());

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"description\"\r\n");
request.writeBytes(fileDescription + "\r\n");

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"file\"; filename=\"" + file.fileName + "\r\n");
request.write(FileUtils.readFileToByteArray(file));
request.writeBytes("\r\n");

request.writeBytes("--" + boundary + "--\r\n");
request.flush();
int respCode = connection.getResponseCode();

switch(respCode) {
    case 200:
        //all went ok - read response
        ...
        break;
    case 301:
    case 302:
}

```

```

    }

//add your file here.
/*This is to add file content*/
for (int i = 0; i < myFileArray.size(); i++) {
    multipart.addFilePart(myFileArray.getParamName(),
        new File(myFileArray.getFileName()));
}

List<String> response = multipart.finish();
Debug.e(TAG, "SERVER REPLIED:");
for (String line : response) {
    Debug.e(TAG, "Upload Files Response:::" + line);
}
// get your server response here.
responseString = line;
}

```

Section 74.7: Upload (POST) file using HttpURLConnection

Quite often it's necessary to send/upload a file to a remote server, for example, an image, video, audio or a backup of the application database to a remote private server. Assuming the server is expecting a POST request with the content, here's a simple example of how to complete this task in Android.

File uploads are sent using multipart/form-data POST requests. It's very easy to implement:

```

URL url = new URL(postTarget);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

String auth = "Bearer " + oauthToken;
connection.setRequestProperty("Authorization", basicAuth);

String boundary = UUID.randomUUID().toString();
connection.setRequestMethod("POST");
connection.setDoOutput(true);
connection.setRequestProperty("Content-Type", "multipart/form-data;boundary=" + boundary);

DataOutputStream request = new DataOutputStream(uc.getOutputStream());

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"description\"\r\n");
request.writeBytes(fileDescription + "\r\n");

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"file\"; filename=\"" + file.fileName + "\r\n");
request.write(FileUtils.readFileToByteArray(file));
request.writeBytes("\r\n");

request.writeBytes("--" + boundary + "--\r\n");
request.flush();
int respCode = connection.getResponseCode();

switch(respCode) {
    case 200:
        //all went ok - read response
        ...
        break;
    case 301:
    case 302:
}

```

```
case 307:  
    //处理重定向 - 例如, 重新发送到新位置  
    ...  
    break;  
...  
default:  
    //执行合理的操作  
}
```

当然，异常需要被捕获或声明抛出。关于这段代码，有几点需要注意：

1. postTarget 是POST的目标URL；oauthToken 是认证令牌；fileDescription 是文件的描述，作为字段 description 的值发送；file 是要发送的文件——它的类型是 `java.io.File` ——如果你有文件路径，可以使用 `new File(filePath)` 来代替。
2. 它为oAuth认证设置Authorization头部
3. 它使用 Apache Common FileUtils 将文件读取为字节数组——如果你已经有文件内容的字节数组或以其他方式存储在内存中，则无需读取文件。
如果你已经有文件内容的字节数组或以其他方式存储在内存中，则无需读取文件。

```
case 307:  
    //handle redirect - for example, re-post to the new location  
    ...  
    break;  
...  
default:  
    //do something sensible  
}
```

Of course, exceptions will need to be caught or declared as being thrown. A couple points to note about this code:

1. postTarget is the destination URL of the POST; oauthToken is the authentication token; fileDescription is the description of the file, which is sent as the value of field description; file is the file to be sent - it's of type `java.io.File` - if you have the file path, you can use `new File(filePath)` instead.
2. It sets Authorization header for an oAuth auth
3. It uses Apache Common FileUtils to read the file into a byte array - if you already have the content of the file in a byte array or in some other way in memory, then there's no need to read it.

第75章：回调URL

第75.1节：使用Instagram OAuth的回调URL示例

回调URL的一个用例是OAuth。让我们用Instagram登录来演示：如果用户输入他们的凭证并点击登录按钮，Instagram将验证凭证并返回一个access_token。我们需要在应用中使用该access_token。

为了让我们的应用能够监听此类链接，我们需要向我们的Activity添加回调URL。我们可以通过向Activity添加一个`<intent-filter>`来实现，该过滤器会响应该回调URL。假设我们的回调URL是`appSchema://appName.com`。那么你需要在`Manifest.xml`文件中向目标Activity添加以下内容：

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:host="appName.com" android:scheme="appSchema" />
```

以上代码行的说明：

- `<category android:name="android.intent.category.BROWSABLE" />` 使目标Activity允许被网页浏览器启动，以显示链接引用的数据。
- `<data android:host="appName.com" android:scheme="appSchema" />` 指定了我们的回调URL的schema和host。
- 总的来说，这些代码行会导致每当浏览器调用回调URL时，特定的Activity被打开。

现在，为了在你的Activity中获取URL的内容，你需要重写`onResume()`方法，代码如下：

```
@Override
public void onResume() {
    // 以下代码行将返回 "appSchema://appName.com".
    String CALLBACK_URL = getResources().getString(R.string.insta_callback);
    Uri uri = getIntent().getData();
    if (uri != null && uri.toString().startsWith(CALLBACK_URL)) {
        String access_token = uri.getQueryParameter("access_token");
    }
    // 在这里执行其他操作。
}
```

现在你已经从Instagram获取了access_token，该令牌用于Instagram的各种API端点。

Chapter 75: Callback URL

Section 75.1: Callback URL example with Instagram OAuth

One of the use cases of *callback URLs* is OAuth. Let us do this with an Instagram Login: If the user enters their credentials and clicks the *Login* button, Instagram will validate the credentials and return an `access_token`. We need that `access_token` in our app.

For our app to be able to listen to such links, we need to add a callback URL to our Activity. We can do this by adding an `<intent-filter>` to our Activity, which will react to that callback URL. Assume that our callback URL is `appSchema://appName.com`. Then you have to add the following lines to your desired Activity in the `Manifest.xml` file:

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:host="appName.com" android:scheme="appSchema" />
```

Explanation of the lines above:

- `<category android:name="android.intent.category.BROWSABLE" />` makes the target activity allow itself to be started by a web browser to display data referenced by a link.
- `<data android:host="appName.com" android:scheme="appSchema" />` specifies our schema and host of our callback URL.
- All together, these lines will cause the specific Activity to be opened whenever the callback URL is called in a browser.

Now, in order to get the contents of the URL in your Activity, you need to override the `onResume()` method as follows:

```
@Override
public void onResume() {
    // The following line will return "appSchema://appName.com".
    String CALLBACK_URL = getResources().getString(R.string.insta_callback);
    Uri uri = getIntent().getData();
    if (uri != null && uri.toString().startsWith(CALLBACK_URL)) {
        String access_token = uri.getQueryParameter("access_token");
    }
    // Perform other operations here.
}
```

Now you have retrieved the `access_token` from Instagram, that is used in various API endpoints of Instagram.

第76章：小吃吧

参数	描述
视图	视图：用于查找父视图的视图。
文本	字符序列：要显示的文本。可以是格式化文本。
资源ID	整数：要使用的字符串资源的资源ID。可以是格式化文本。
持续时间	整数：消息显示的时长。可以是 LENGTH_SHORT、LENGTH_LONG 或 LENGTH_INDEFINITE

第76.1节：创建一个简单的Snackbar

创建一个Snackbar可以按如下方式进行：

```
Snackbar.make(view, "要显示的文本", Snackbar.LENGTH_LONG).show();
```

view用于查找合适的父视图来显示Snackbar。通常这会是你在XML中定义的CoordinatorLayout，它支持添加诸如滑动关闭和自动移动其他控件（例如FloatingActionButton）等功能。如果没有CoordinatorLayout，则使用窗口装饰的内容视图。

我们经常还会给Snackbar添加一个操作。一个常见的用例是“撤销”操作。

```
Snackbar.make(view, "要显示的文本", Snackbar.LENGTH_LONG)
    .setAction("UNDO", new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // 在这里编写你的逻辑
        }
    })
.show();
```

你可以创建一个Snackbar并稍后显示：

```
Snackbar snackbar = Snackbar.make(view, "要显示的文本", Snackbar.LENGTH_LONG);
snackbar.show();
```

如果你想改变Snackbar文本的颜色：

```
Snackbar snackbar = Snackbar.make(view, "显示的文本", Snackbar.LENGTH_LONG);
View view = snackbar.getView();
TextView textView = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.parseColor("#FF4500"));
snackbar.show();
```

默认情况下，Snackbar 会在向右滑动时消失。此示例演示如何在向左滑动时

[关闭 Snackbar](#)。

第76.2节：自定义 Snackbar

自定义 Snackbar 的函数

```
public static Snackbar makeText(Context context, String message, int duration) {
    Activity activity = (Activity) context;
```

Chapter 76: Snackbar

Parameter	Description
view	View: The view to find a parent from.
text	CharSequence: The text to show. Can be formatted text.
resId	int: The resource id of the string resource to use. Can be formatted text.
duration	int: How long to display the message. This can be LENGTH_SHORT, LENGTH_LONG or LENGTH_INDEFINITE

Section 76.1: Creating a simple Snackbar

Creating a Snackbar can be done as follows:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

The view is used to find a suitable parent to use to display the Snackbar. Typically this would be a CoordinatorLayout that you've defined in your XML, which enables adding functionality such as swipe to dismiss and automatically moving of other widgets (e.g. FloatingActionButton). If there's no CoordinatorLayout then the window decor's content view is used.

Very often we also add an action to the Snackbar. A common use case would be an "Undo" action.

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG)
    .setAction("UNDO", new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // put your logic here
        }
    })
.show();
```

You can create a Snackbar and show it later:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

If you want to change the color of the Snackbar's text:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
View view = snackbar.getView();
TextView textView = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.parseColor("#FF4500"));
snackbar.show();
```

By default Snackbar dismisses on its right swipe. This example demonstrates how to [dismiss the snackBar on its left swipe](#).

Section 76.2: Custom Snack Bar

Function to customize snack bar

```
public static Snackbar makeText(Context context, String message, int duration) {
    Activity activity = (Activity) context;
```

```

View layout;
Snackbar snackbar = Snackbar
    .make(activity.findViewById(android.R.id.content), message, duration);
    layout = snackbar.getView();
    //设置背景颜色
layout.setBackgroundColor(context.getResources().getColor(R.color.orange));
    android.widget.TextView text = (android.widget.TextView)
layout.findViewById(android.support.design.R.id.snackbar_text);
    //设置字体颜色
text.setTextColor(context.getResources().getColor(R.color.white));
    Typeface font = null;
    //设置字体
font = Typeface.createFromAsset(context.getAssets(), "DroidSansFallbackanmol256.ttf");
    text.setTypeface(font);
    return snackbar;
}

```

从片段或活动中调用该函数

```

SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();

```

第76.3节：自定义Snackbar（无需视图）

创建一个Snackbar，无需传入视图，所有布局都创建在android.R.id.content中。

```

public class CustomSnackBar {

    public static final int STATE_ERROR = 0;
    public static final int STATE_WARNING = 1;
    public static final int STATE_SUCCESS = 2;
    public static final int VIEW_PARENT = android.R.id.content;

    public CustomSnackBar(View view, String message, int actionType) {
        super();

        Snackbar snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
            textView.setTextColor(Color.parseColor("#ffffffff"));
            textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
            textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
            textView.setLayoutDirection(View.LAYOUT_DIRECTION_RTL);

            switch (actionType) {
                case STATE_ERROR:
snackbar.getView().setBackgroundColor(Color.parseColor("#F12B2B"));
                    break;
                case STATE_WARNING:
snackbar.getView().setBackgroundColor(Color.parseColor("#000000"));
                    break;
                case STATE_SUCCESS:
snackbar.getView().setBackgroundColor(Color.parseColor("#7ED321"));
                    break;
            }
        snackbar.show();
    }
}

```

```

View layout;
Snackbar snackbar = Snackbar
    .make(activity.findViewById(android.R.id.content), message, duration);
    layout = snackbar.getView();
    //setting background color
layout.setBackgroundColor(context.getResources().getColor(R.color.orange));
    android.widget.TextView text = (android.widget.TextView)
layout.findViewById(android.support.design.R.id.snackbar_text);
    //setting font color
text.setTextColor(context.getResources().getColor(R.color.white));
    Typeface font = null;
    //Setting font
font = Typeface.createFromAsset(context.getAssets(), "DroidSansFallbackanmol256.ttf");
    text.setTypeface(font);
    return snackbar;
}

```

Call the function from fragment or activity

```

SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();

```

Section 76.3: Custom Snackbar (no need view)

Creating an Snackbar without the need pass view to Snackbar, all layout create in android.R.id.content.

```

public class CustomSnackBar {

    public static final int STATE_ERROR = 0;
    public static final int STATE_WARNING = 1;
    public static final int STATE_SUCCESS = 2;
    public static final int VIEW_PARENT = android.R.id.content;

    public CustomSnackBar(View view, String message, int actionType) {
        super();

        Snackbar snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
            textView.setTextColor(Color.parseColor("#ffffffff"));
            textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
            textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
            textView.setLayoutDirection(View.LAYOUT_DIRECTION_RTL);

            switch (actionType) {
                case STATE_ERROR:
snackbar.getView().setBackgroundColor(Color.parseColor("#F12B2B"));
                    break;
                case STATE_WARNING:
snackbar.getView().setBackgroundColor(Color.parseColor("#000000"));
                    break;
                case STATE_SUCCESS:
snackbar.getView().setBackgroundColor(Color.parseColor("#7ED321"));
                    break;
            }
        snackbar.show();
    }
}

```

```
new CustomSnackBar(findViewById(CustomSnackBar.VIEW_PARENT), "message", CustomSnackBar.STATE_ERROR);
```

第76.4节：带回调的Snackbar

您可以使用Snackbar.Callback来监听Snackbar是否被用户或超时关闭。

```
Snackbar.make(getView(), "Hi Snackbar!", Snackbar.LENGTH_LONG).setCallback( new Snackbar.Callback() {
    @Override
    public void onDismissed(Snackbar snackbar, int event) {
        switch(event) {
            case Snackbar.Callback.DISMISS_EVENT_ACTION:
                Toast.makeText(getApplicationContext(), "点击了操作",
                Toast.LENGTH_LONG).show();
                break;
            case Snackbar.Callback.DISMISS_EVENT_TIMEOUT:
                Toast.makeText(getApplicationContext(), "超时", Toast.LENGTH_LONG).show();
                break;
        }
    }

    @Override
    public void onShown(Snackbar snackbar) {
        Toast.makeText(getApplicationContext(), "这是我讨厌的继兄",
        Toast.LENGTH_LONG).show();
    }
}).setAction("去！", new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ...
    }
}).show();
```

第76.5节：Snackbar与Toast：我应该使用哪一个？

当我们想向用户显示某个操作已成功（或未成功）发生的信息，且该操作不需要用户采取其他行动时，通常使用Toast。比如当消息已发送时：

```
Toast.makeText(this, "消息已发送！", Toast.LENGTH_SHORT).show();
```

Snackbars 也用于显示信息。但这次，我们可以给用户一个执行操作的机会。例如，假设用户错误地删除了一张图片，他想要恢复它。我们可以提供一个带有“撤销”操作的 Snackbar。像这样：

```
Snackbar.make(getApplicationContext(), "图片已删除", Snackbar.LENGTH_SHORT)
.setAction("撤销", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //恢复他的图片
    }
})
.show();
```

结论：当我们不需要用户交互时使用 Toasts。Snackbars 用于允许用户执行

for call class

```
new CustomSnackBar(findViewById(CustomSnackBar.VIEW_PARENT), "message", CustomSnackBar.STATE_ERROR);
```

Section 76.4: Snackbar with Callback

You can use Snackbar.Callback to listen if the Snackbar was dismissed by user or timeout.

```
Snackbar.make(getView(), "Hi Snackbar!", Snackbar.LENGTH_LONG).setCallback( new Snackbar.Callback() {
    @Override
    public void onDismissed(Snackbar snackbar, int event) {
        switch(event) {
            case Snackbar.Callback.DISMISS_EVENT_ACTION:
                Toast.makeText(getApplicationContext(), "Clicked the action",
                Toast.LENGTH_LONG).show();
                break;
            case Snackbar.Callback.DISMISS_EVENT_TIMEOUT:
                Toast.makeText(getApplicationContext(), "Time out", Toast.LENGTH_LONG).show();
                break;
        }
    }

    @Override
    public void onShown(Snackbar snackbar) {
        Toast.makeText(getApplicationContext(), "This is my annoying step-brother",
        Toast.LENGTH_LONG).show();
    }
}).setAction("Go!", new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ...
    }
}).show();
```

Section 76.5: Snackbar vs Toasts: Which one should I use?

Toasts are generally used when we want to display an information to the user regarding some action that has successfully (or not) happened and this action does not require the user to take any other action. Like when a message has been sent, for example:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Snackbars are also used to display an information. But this time, we can give the user an opportunity to take an action. For example, let's say the user deleted a picture by mistake and he wants to get it back. We can provide a Snackbar with the "Undo" action. Like this:

```
Snackbar.make(getApplicationContext(), "Picture Deleted", Snackbar.LENGTH_SHORT)
.setAction("Undo", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Return his picture
    }
})
.show();
```

Conclusion: Toasts are used when we don't need user interaction. Snackbars are used to allow users to take

另一个操作或撤销之前的操作。

第 76.6 节：自定义 Snackbar

此示例展示了一个带有自定义撤销图标的白色 Snackbar。

```
Snackbar customBar = Snackbar.make(view, "要显示的文本", Snackbar.LENGTH_LONG);
customBar.setAction("撤销", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //在这里放置撤销按钮的逻辑
    }
});

View sbView = customBar.getView();
//将背景更改为白色
sbView.setBackgroundColor(Color.WHITE);

TextView snackText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_text);
if (snackText!=null) {
    //将文本颜色更改为黑色
    snackText.setTextColor(Color.BLACK);
}

TextView actionText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_action);
if (actionText!=null) {
    // 设置自定义撤销图标
    actionText.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.custom_undo, 0, 0, 0);
}
customBar.show();
```

another action or undo a previous one.

Section 76.6: Custom Snackbar

This example shows a white Snackbar with custom Undo icon.

```
Snackbar customBar = Snackbar.make(view, "Text to be displayed", Snackbar.LENGTH_LONG);
customBar.setAction("UNDO", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Put the logic for undo button here
    }
});

View sbView = customBar.getView();
//Changing background to White
sbView.setBackgroundColor(Color.WHITE);

TextView snackText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_text);
if (snackText!=null) {
    //Changing text color to Black
    snackText.setTextColor(Color.BLACK);
}

TextView actionText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_action);
if (actionText!=null) {
    // Setting custom Undo icon
    actionText.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.custom_undo, 0, 0, 0);
}
customBar.show();
```

第77章：控件

第77.1节：清单声明 -

在应用程序的AndroidManifest.xml文件中声明AppWidgetProvider类。例如：

```
<receiver android:name="ExampleAppWidgetProvider" >
<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>
<meta-data android:name="android.appwidget.provider"
    android:resource="@xml/example_appwidget_info" />
</receiver>
```

第77.2节：元数据

在res/xml中添加AppWidgetProviderInfo元数据：

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen">
</appwidget-provider>
```

第77.3节：AppWidgetProvider类

最重要的AppWidgetProvider回调是onUpdate()。每次添加appwidget时都会调用它。

```
public class ExampleAppWidgetProvider extends AppWidgetProvider {

    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        final int N = appWidgetIds.length;

        // 对属于此提供者的每个App Widget执行此循环过程
        for (int i=0; i<N; i++) {
            int appWidgetId = appWidgetIds[i];

            // 创建一个Intent以启动ExampleActivity
            Intent intent = new Intent(context, ExampleActivity.class);
            PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);

            // 获取App Widget的布局并为按钮附加点击监听器
            //
            RemoteViews views = new RemoteViews(context.getPackageName(),
                R.layout.appwidget_provider_layout);
            views.setOnClickPendingIntent(R.id.button, pendingIntent);

            // 告诉 AppWidgetManager 对当前应用小部件执行更新
            appWidgetManager.updateAppWidget(appWidgetId, views);
        }
    }
}
```

Chapter 77: Widgets

Section 77.1: Manifest Declaration -

Declare the AppWidgetProvider class in your application's AndroidManifest.xml file. For example:

```
<receiver android:name="ExampleAppWidgetProvider" >
<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>
<meta-data android:name="android.appwidget.provider"
    android:resource="@xml/example_appwidget_info" />
</receiver>
```

Section 77.2: Metadata

Add the AppWidgetProviderInfo metadata in res/xml:

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen">
</appwidget-provider>
```

Section 77.3: AppWidgetProvider Class

The most important AppWidgetProvider callback is `onUpdate()`. It is called every time an appwidget is added.

```
public class ExampleAppWidgetProvider extends AppWidgetProvider {

    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        final int N = appWidgetIds.length;

        // Perform this loop procedure for each App Widget that belongs to this provider
        for (int i=0; i<N; i++) {
            int appWidgetId = appWidgetIds[i];

            // Create an Intent to launch ExampleActivity
            Intent intent = new Intent(context, ExampleActivity.class);
            PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);

            // Get the layout for the App Widget and attach an on-click listener
            // to the button
            RemoteViews views = new RemoteViews(context.getPackageName(),
                R.layout.appwidget_provider_layout);
            views.setOnClickListener(R.id.button, pendingIntent);

            // Tell the AppWidgetManager to perform an update on the current app widget
            appWidgetManager.updateAppWidget(appWidgetId, views);
        }
    }
}
```

onAppWidgetOptionsChanged() 在小部件被放置或调整大小时调用。

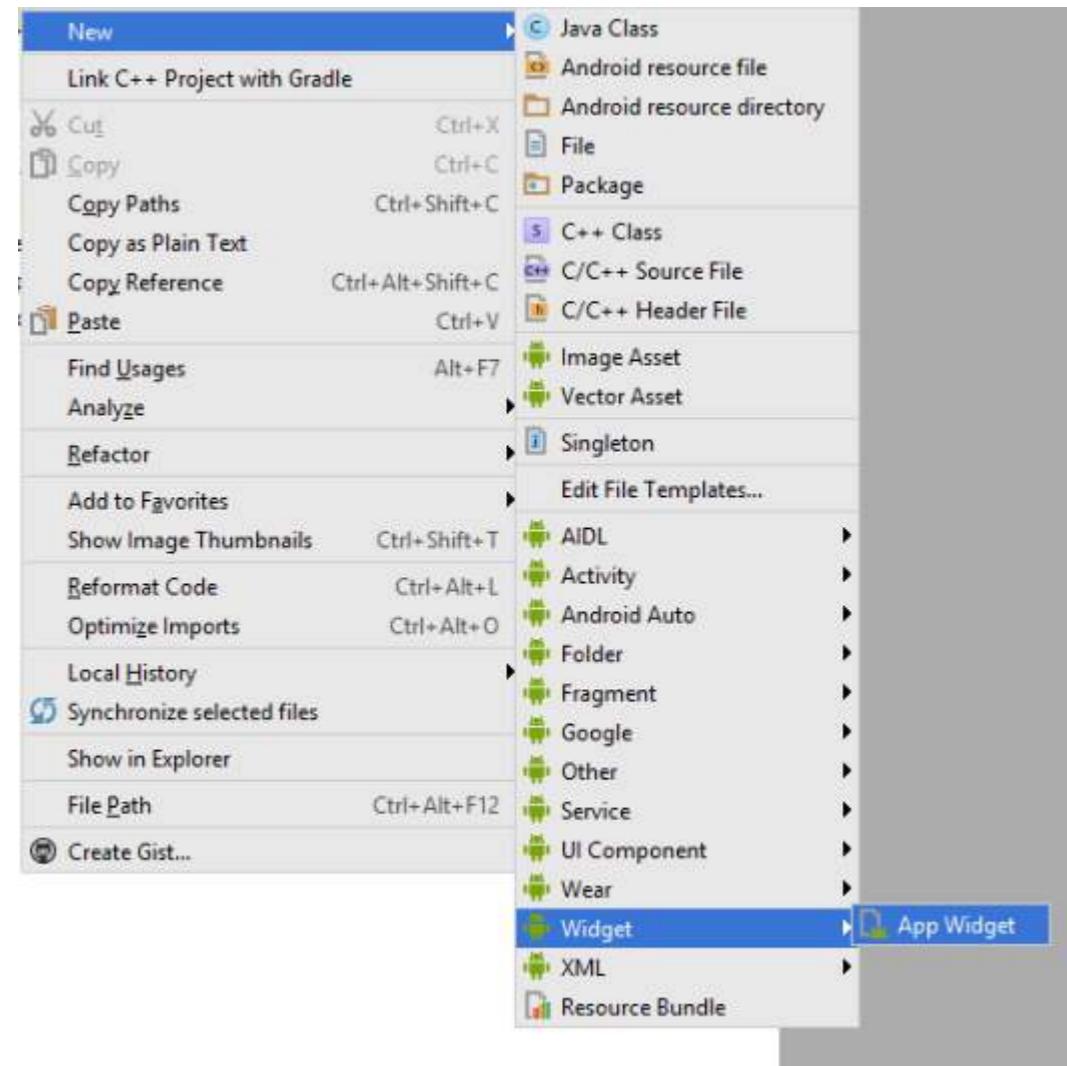
onDeleted(Context, int[]) 在小部件被删除时调用。

第77.4节：使用 Android

Studio 创建/集成基础小部件

最新的 Android Studio 将通过两步将基础小部件创建并集成到您的应用中。

直接在您的应用中 ==> 新建 ==> 小部件 ==> 应用小部件



将显示如下屏幕并填写字段

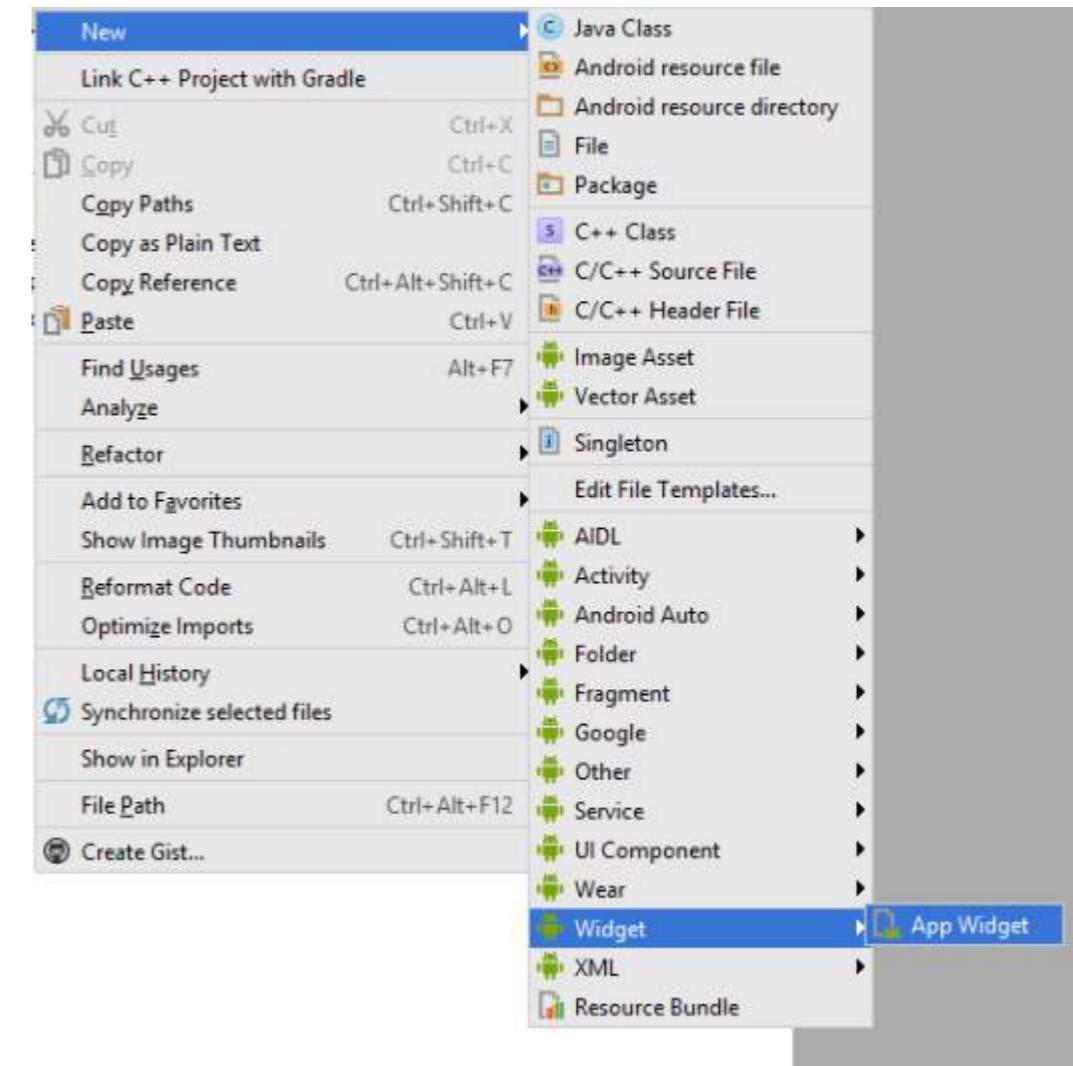
onAppWidgetOptionsChanged() is called when the widget is placed or resized.

onDeleted(Context, int[]) is called when the widget is deleted.

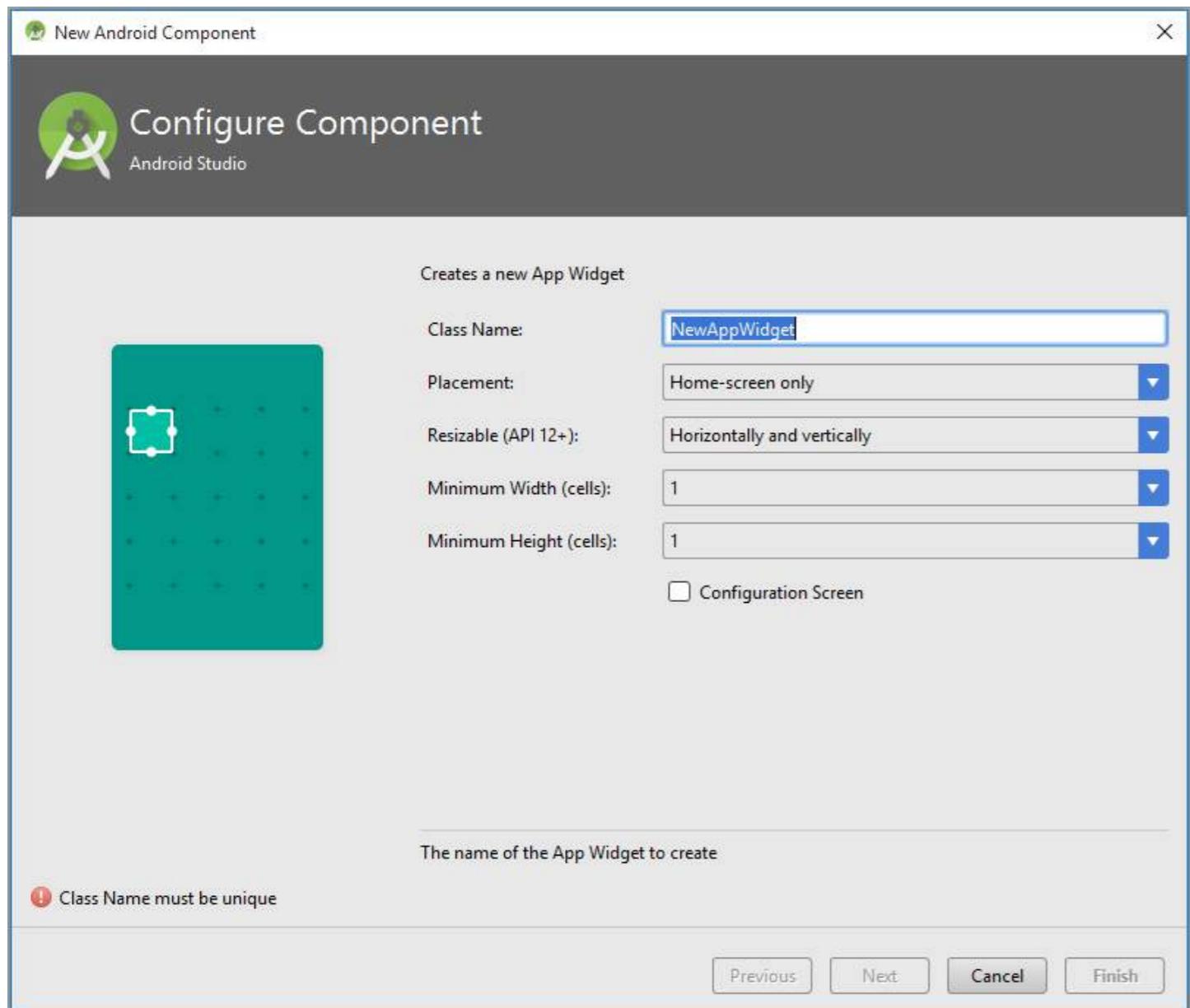
Section 77.4: Create/Integrate Basic Widget using Android Studio

Latest Android Studio will create & integrate a Basic Widget to your Application in 2 steps.

Right on your Application ==> New ==> Widget ==> App Widget



It will show a Screen like below & fill the fields



完成。

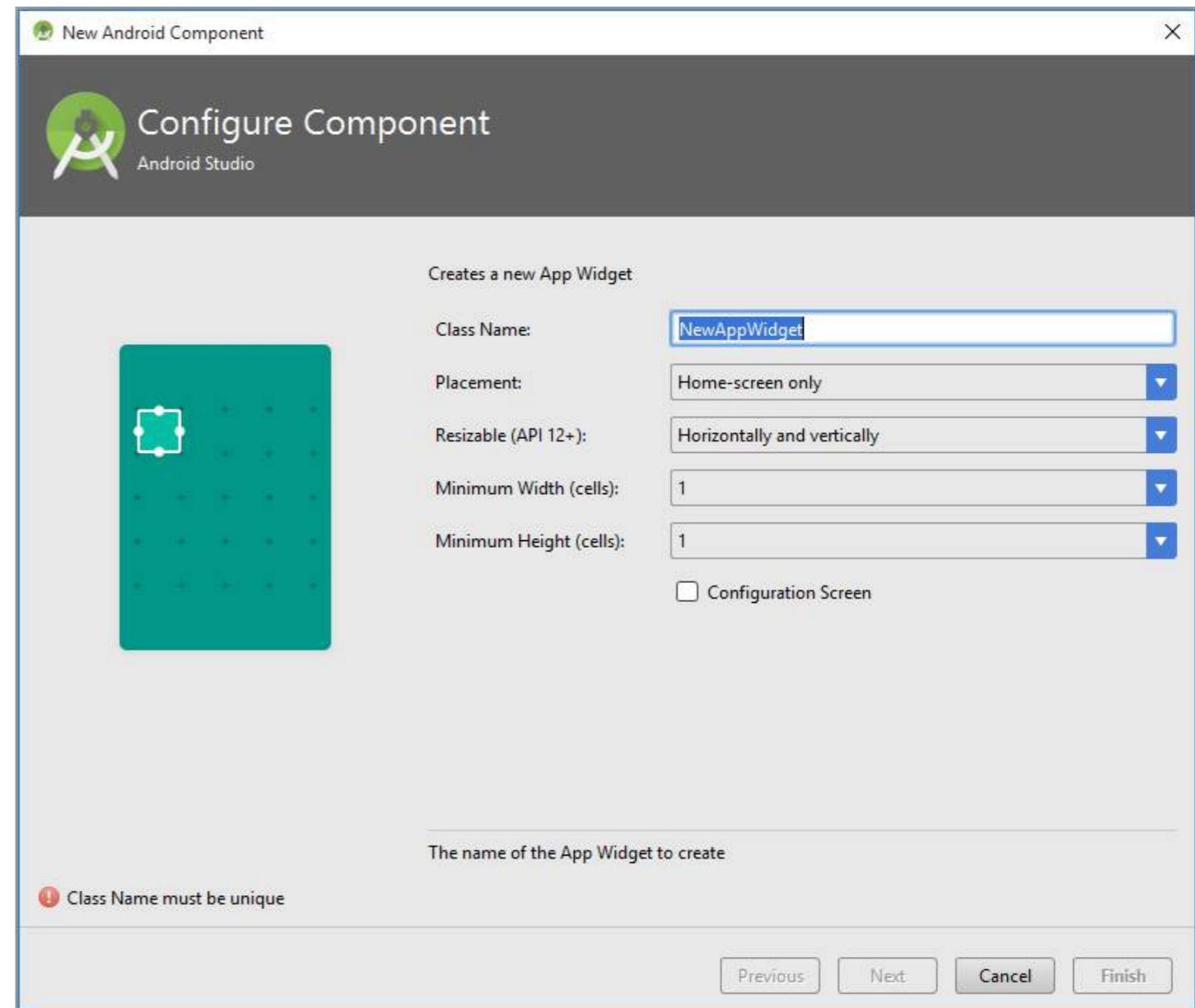
它将创建并集成一个基本的HelloWorld小部件（包括布局文件、元数据文件、清单文件中的声明等）到您的应用程序中。

第77.5节：两个具有不同布局声明的小部件

- 在清单文件中声明两个接收器：

```
<receiver
    android:name=".UVMateWidget"
    android:label="UVMate Widget 1x1">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_1x1" />
</receiver>
<receiver
    android:name=".UVMateWidget2x2"
    android:label="UVMate Widget 2x2">
```



Its Done.

It will **create & integrate a basic HelloWorld Widget**(Including Layout File , Meta Data File , Declaration in Manifest File etc.) to your Application.

Section 77.5: Two widgets with different layouts declaration

- Declare two receivers in a manifest file:

```
<receiver
    android:name=".UVMateWidget"
    android:label="UVMate Widget 1x1">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_1x1" />
</receiver>
<receiver
    android:name=".UVMateWidget2x2"
    android:label="UVMate Widget 2x2">
```

```

<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>

<meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/widget_2x2" />
</receiver>

```

2. 创建两个布局

- @xml/widget_1x1
- @xml/widget_2x2

3. 声明继承自UVMateWidget类并具有扩展行为的子类UVMateWidget2x2：

```

package au.com.aershov.uvmate;

import android.content.Context;
import android.widget.RemoteViews;

public class UVMateWidget2x2 extends UVMateWidget {

    public RemoteViews getRemoteViews(Context context, int minWidth,
                                      int minHeight) {

        mUVMateHelper.saveWidgetSize(mContext.getString(R.string.app_ws_2x2));
        return new RemoteViews(context.getPackageName(), R.layout.widget_2x2);
    }
}

```

```

<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>

<meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/widget_2x2" />
</receiver>

```

2. Create two layouts

- @xml/widget_1x1
- @xml/widget_2x2

3. Declare the subclass UVMateWidget2x2 from the UVMateWidget class with extended behavior:

```

package au.com.aershov.uvmate;

import android.content.Context;
import android.widget.RemoteViews;

public class UVMateWidget2x2 extends UVMateWidget {

    public RemoteViews getRemoteViews(Context context, int minWidth,
                                      int minHeight) {

        mUVMateHelper.saveWidgetSize(mContext.getString(R.string.app_ws_2x2));
        return new RemoteViews(context.getPackageName(), R.layout.widget_2x2);
    }
}

```

第78章：Toast

参数

	详细信息
上下文	用于显示Toast的上下文。this通常用于Activity中，getActivity()通常用于Fragment中
文本	指定Toast中显示文本的CharSequence。任何实现了CharSequence的对象都可以使用，包括字符串
资源ID	可用于提供资源字符串以显示在Toast中的资源ID
持续时间	表示Toast显示时长的整数标志。选项有Toast.LENGTH_SHORT和Toast.LENGTH_LONG
重力	指定Toast位置或“重力”的整数。选项见 here
x偏移量	指定Toast位置的水平偏移量
yOffset	指定 Toast 位置的垂直偏移量

一个Toast在一个小弹出窗口中提供关于操作的简单反馈，并在超时后自动消失。它只占用消息所需的空间，当前活动仍然可见且可交互。

第78.1节：创建自定义 Toast

如果您不想使用默认的 Toast 视图，可以通过setView(View)方法为Toast对象提供自定义视图。

首先，创建您想在 Toast 中使用的 XML 布局。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="8dp"
    android:background="#111">

    <TextView android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF" />

    <TextView android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF" />

</LinearLayout>
```

然后，在创建你的Toast时，从XML中加载自定义视图，并调用setView

```
// 从XML加载自定义视图
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast_layout,
    (ViewGroup) findViewById(R.id.toast_layout_root));

// 设置自定义布局中的标题和描述TextView
TextView title = (TextView) layout.findViewById(R.id.title);
title.setText("Toast标题");
```

Chapter 78: Toast

Parameter

	Details
context	The context to display your Toast in. this is commonly used in an Activity and getActivity() is commonly used in a Fragment
text	A CharSequence that specifies what text will be shown in the Toast. Any object that implements CharSequence can be used, including a String
resId	A resource ID that can be used to provide a resource String to display in the Toast
duration	Integer flag representing how long the Toast will show. Options are Toast.LENGTH_SHORT and Toast.LENGTH_LONG
gravity	Integer specifying the position, or "gravity" of the Toast. See options here
xOffset	Specifies the horizontal offset for the Toast position
yOffset	Specifies the vertical offset for the Toast position

A [Toast](#) provides simple feedback about an operation in a small popup and automatically disappears after a timeout. It only fills the amount of space required for the message and the current activity remains visible and interactive.

Section 78.1: Creating a custom Toast

If you don't want to use the default Toast view, you can provide your own using the setView([View](#)) method on a Toast object.

First, create the XML layout you would like to use in your Toast.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="8dp"
    android:background="#111">

    <TextView android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF" />

    <TextView android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF" />

</LinearLayout>
```

Then, when creating your Toast, inflate your custom View from XML, and call setView

```
// Inflate the custom view from XML
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast_layout,
    (ViewGroup) findViewById(R.id.toast_layout_root));

// Set the title and description TextViews from our custom layout
TextView title = (TextView) layout.findViewById(R.id.title);
title.setText("Toast Title");
```

```
TextView description = (TextView) layout.findViewById(R.id.description);
description.setText("Toast描述");

// 创建并显示Toast对象

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

第78.2节：设置Toast的位置

标准的Toast通知会出现在屏幕底部，水平居中对齐。你可以通过setGravity(int, int, int)来改变这个位置。该方法接受三个参数：一个Gravity常量、一个x轴偏移量和一个y轴偏移量。

例如，如果你想让Toast出现在左上角，可以这样设置gravity：

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

第78.3节：显示Toast消息

在Android中，Toast是一个简单的UI元素，可以用来向用户提供上下文反馈。

要显示一个简单的Toast消息，我们可以这样做。

```
// 声明用于Toast的参数

Context context = getApplicationContext();
// 在一个Activity中，你也可以使用 "this"
// 在一个Fragment中，你可以使用 getActivity()

CharSequence message = "我是一个 Android Toast!";
int duration = Toast.LENGTH_LONG; // 另一个选项是 Toast.LENGTH_SHORT

// 创建 Toast 对象，并显示它！
Toast myToast = Toast.makeText(context, message, duration);
myToast.show();
```

或者，要内联显示 Toast，而不保存 Toast 对象，可以：

```
Toast.makeText(context, "叮！你的 Toast 已准备好。", Toast.LENGTH_SHORT).show();
```

重要提示：确保 show() 方法是在 UI 线程中调用的。如果你尝试从不同线程显示 Toast，可以例如使用 Activity 的 runOnUiThread 方法。

如果不这样做，即尝试通过创建 Toast 来修改 UI，将抛出一个 RuntimeException，错误信息类似如下：

```
java.lang.RuntimeException: 无法在未调用 Looper.prepare() 的线程内创建 handler
```

处理此异常的最简单方法是使用 runOnUiThread：语法如下所示。

```
runOnUiThread(new Runnable() {
    @Override
```

```
TextView description = (TextView) layout.findViewById(R.id.description);
description.setText("Toast Description");

// Create and show the Toast object

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

Section 78.2: Set position of a Toast

A standard toast notification appears at the bottom of the screen aligned in horizontal centre. You can change this position with the setGravity(**int, int, int**). This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

For example, if you decide that the toast should appear in the top-left corner, you can set the gravity like this:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Section 78.3: Showing a Toast Message

In Android, a Toast is a simple UI element that can be used to give contextual feedback to a user.

To display a simple Toast message, we can do the following.

```
// Declare the parameters to use for the Toast

Context context = getApplicationContext();
// in an Activity, you may also use "this"
// in a fragment, you can use getActivity()

CharSequence message = "I'm an Android Toast!";
int duration = Toast.LENGTH_LONG; // Toast.LENGTH_SHORT is the other option

// Create the Toast object, and show it!
Toast myToast = Toast.makeText(context, message, duration);
myToast.show();
```

Or, to show a Toast inline, without holding on to the Toast object you can:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

IMPORTANT: Make sure that the `show()` method is called from the UI thread. If you're trying to show a Toast from a different thread you can e.g. use `runOnUiThread` method of an `Activity`.

Failing to do so, meaning trying to modify the UI by creating a Toast, will throw a `RuntimeException` which will look like this:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called Looper.prepare()
```

The simplest way of handling this exception is just by using `runOnUiThread`: syntax is shown below.

```
runOnUiThread(new Runnable() {
    @Override
```

```
public void run() {
    // 你的代码在这里
}
});
```

第78.4节：在软键盘上方显示Toast消息

默认情况下，即使键盘显示，Android也会在屏幕底部显示Toast消息。这将会在键盘正上方显示Toast消息。

```
public void showMessage(final String message, final int length) {
    View root = findViewById(android.R.id.content);
    Toast toast = Toast.makeText(this, message, length);
    int yOffset = Math.max(0, root.getHeight() - toast.getYOffset());
    toast.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL, 0, yOffset);
    toast.show();
}
```

第78.5节：线程安全的Toast显示方式 (应用范围内)

```
public class MainApplication extends Application {

    private static Context context; // 应用程序上下文

    private Handler mainThreadHandler;
    private Toast toast;

    public Handler getMainThreadHandler() {
        if (mainThreadHandler == null) {
            mainThreadHandler = new Handler(Looper.getMainLooper());
        }
        return mainThreadHandler;
    }

    @Override public void onCreate() {
        super.onCreate();
        context = this;
    }

    public static MainApplication getApp(){
        return (MainApplication) context;
    }

    /**
     * 线程安全的显示吐司方法。
     * @param message
     * @param duration
     */
    public void showToast(final String message, final int duration) {
        getMainThreadHandler().post(new Runnable() {
            @Override
            public void run() {
                if (!TextUtils.isEmpty(message)) {
                    if (toast != null) {
                        toast.cancel(); //如果当前吐司可见，则取消显示
                        toast.setText(message);
                    } else {
                        toast = Toast.makeText(App.this, message, duration);
                    }
                }
            }
        });
    }
}
```

```
public void run() {
    // Your code here
}
});
```

Section 78.4: Show Toast Message Above Soft Keyboard

By default, Android will display Toast messages at the bottom of the screen even if the keyboard is showing. This will show a Toast message just above the keyboard.

```
public void showMessage(final String message, final int length) {
    View root = findViewById(android.R.id.content);
    Toast toast = Toast.makeText(this, message, length);
    int yOffset = Math.max(0, root.getHeight() - toast.getYOffset());
    toast.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL, 0, yOffset);
    toast.show();
}
```

Section 78.5: Thread safe way of displaying Toast (Application Wide)

```
public class MainApplication extends Application {

    private static Context context; //application context

    private Handler mainThreadHandler;
    private Toast toast;

    public Handler getMainThreadHandler() {
        if (mainThreadHandler == null) {
            mainThreadHandler = new Handler(Looper.getMainLooper());
        }
        return mainThreadHandler;
    }

    @Override public void onCreate() {
        super.onCreate();
        context = this;
    }

    public static MainApplication getApp(){
        return (MainApplication) context;
    }

    /**
     * Thread safe way of displaying toast.
     * @param message
     * @param duration
     */
    public void showToast(final String message, final int duration) {
        getMainThreadHandler().post(new Runnable() {
            @Override
            public void run() {
                if (!TextUtils.isEmpty(message)) {
                    if (toast != null) {
                        toast.cancel(); //dismiss current toast if visible
                        toast.setText(message);
                    } else {
                        toast = Toast.makeText(App.this, message, duration);
                    }
                }
            }
        });
    }
}
```

```
        }
        toast.show();
    }
});
```

记得在manifest中添加MainApplication。

现在可以从任何线程调用它来显示吐司消息。

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

第78.6节：线程安全的显示吐司消息方式 (针对AsyncTask)

如果你不想继承Application且想保持吐司消息的线程安全，确保在AsyncTask的post execute部分显示它们。

```
public class MyAsyncTask extends AsyncTask <Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... params) {
        // 在这里执行你的后台工作
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        // 在这里显示吐司消息
        Toast.makeText(context, "叮！您的吐司已准备好。", Toast.LENGTH_SHORT).show();
    }
}
```

```
        }
        toast.show();
    }
});
```

Remember to add MainApplication in manifest.

Now call it from any thread to display a toast message.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Section 78.6: Thread safe way of displaying a Toast Message (For AsyncTask)

If you don't want to extend Application and keep your toast messages thread safe, make sure you show them in the post execute section of your AsyncTasks.

```
public class MyAsyncTask extends AsyncTask <Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... params) {
        // Do your background work here
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        // Show toast messages here
        Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
    }
}
```

第79章：为吐司消息创建单例类

参数

	详情
上下文	需要显示吐司消息的相关上下文。如果在活动中使用，请传入 "this" 关键字；如果在片段中使用，请传入 "getActivity()"。
视图	创建一个自定义视图并将该视图对象传递给此方法。
重力	传递吐司的位置重力。所有位置都作为静态变量添加在Gravity类下。最常用的位置有Gravity.TOP、Gravity.BOTTOM、Gravity.LEFT、Gravity.RIGHT。
x偏移量	吐司消息的水平偏移量。
yOffset	吐司消息的垂直偏移量。
持续时间	吐司显示的持续时间。我们可以设置为Toast.LENGTH_SHORT或Toast.LENGTH_LONG

Toast 消息是向用户提供反馈的最简单方式。默认情况下，Android 提供灰色的消息 Toast，我们可以设置消息内容和显示时长。如果需要创建更可定制且可重用的 Toast 消息，可以通过自定义布局自行实现。更重要的是，在实现时，使用单例设计模式将使自定义 Toast 消息类的维护和开发更加方便。

第 79.1 节：为 Toast 消息创建自己的单例类

以下是如何为 Toast 消息创建自己的单例类，如果您的应用需要针对不同用例显示成功、警告和危险消息，您可以在根据自己的需求修改后使用此类。

```
public class ToastGenerate {
    private static ToastGenerate ourInstance;

    public ToastGenerate (Context context) {
        this.context = context;
    }

    public static ToastGenerate getInstance(Context context) {
        if (ourInstance == null)
            ourInstance = new ToastGenerate(context);
        return ourInstance;
    }

    //将消息和消息类型传递给此方法
    public void createToastMessage(String message,int type){

        //加载自定义布局
        LayoutInflator layoutInflater = (LayoutInflator)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        LinearLayout toastLayout = (LinearLayout)
        layoutInflater.inflate(R.layout.layout_custum_toast,null);
        TextView toastShowMessage = (TextView)
        toastLayout.findViewById(R.id.textCustomToastTopic);

        switch (type){
            case 0:
                //如果消息类型是0，将调用失败提示方法
                createFailToast(toastLayout,toastShowMessage,message);
                break;
            case 1:
        }
    }
}
```

Chapter 79: Create Singleton Class for Toast Message

Parameter

	details
context	Relevant context which needs to display your toast message. If you use this in the activity pass "this" keyword or If you use in fragment pass as "getActivity()".
view	Create a custom view and pass that view object to this.
gravity	Pass the gravity position of the toaster. All the position has added under the Gravity class as the static variables . The Most common positions are Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Horizontal offset of the toast message.
yOffset	Vertical offset of the toast message.
duration	Duration of the toast show. We can set either Toast.LENGTH_SHORT or Toast.LENGTH_LONG

Toast messages are the most simple way of providing feedback to the user. By default, Android provide gray color message toast where we can set the message and the duration of the message. If we need to create more customizable and reusable toast message, we can implement it by ourselves with the use of a custom layout. More importantly when we are implementing it, the use of Singleton design pattern will make it easy for maintaining and development of the custom toast message class.

Section 79.1: Create own singleton class for toast massages

Here is how to create your own singleton class for toast messages, If your application need to show success, warning and the danger messages for different use cases you can use this class after you have modified it to your own specifications.

```
public class ToastGenerate {
    private static ToastGenerate ourInstance;

    public ToastGenerate (Context context) {
        this.context = context;
    }

    public static ToastGenerate getInstance(Context context) {
        if (ourInstance == null)
            ourInstance = new ToastGenerate(context);
        return ourInstance;
    }

    //pass message and message type to this method
    public void createToastMessage(String message,int type){

        //inflate the custom layout
        LayoutInflator layoutInflater = (LayoutInflator)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        LinearLayout toastLayout = (LinearLayout)
        layoutInflater.inflate(R.layout.layout_custum_toast,null);
        TextView toastShowMessage = (TextView)
        toastLayout.findViewById(R.id.textCustomToastTopic);

        switch (type){
            case 0:
                //if the message type is 0 fail toaster method will call
                createFailToast(toastLayout,toastShowMessage,message);
                break;
            case 1:
        }
    }
}
```

```

        //如果消息类型是1, 将调用成功提示方法
createSuccessToast(toastLayout,toastShowMessage,message);
    break;

    case 2:
createWarningToast( toastLayout, toastShowMessage, message);
        //如果消息类型是2, 将调用警告提示方法
    break;
default:
createFailToast(toastLayout,toastShowMessage,message);

}

//失败提示消息方法
private final void createFailToast(LinearLayout toastLayout,TextView toastMessage,String message){

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.button_alert_normal));
    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context,toastLayout);
}

//警告提示消息方法
private final void createWarningToast( LinearLayout toastLayout, TextView toastMessage,
String message) {
toastLayout.setBackgroundColor(context.getResources().getColor(R.color.warning_toast));
    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context, toastLayout);
}

//成功提示消息方法
private final void createSuccessToast(LinearLayout toastLayout,TextView toastMessage,String message){
toastLayout.setBackgroundColor(context.getResources().getColor(R.color.success_toast));

    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context,toastLayout);
}

private void showToast(View view){
Toast toast = new Toast(context);
    toast.setGravity(Gravity.TOP,0,0); // 在设备顶部显示消息
    toast.setDuration(Toast.LENGTH_SHORT);
toast.setView(view);
    toast.show();
}
}

```

```

        //if the message type is 1 success toaster method will call
createSuccessToast(toastLayout,toastShowMessage,message);
    break;

    case 2:
createWarningToast( toastLayout, toastShowMessage, message);
        //if the message type is 2 warning toaster method will call
    break;
default:
createFailToast(toastLayout,toastShowMessage,message);

}

//Failure toast message method
private final void createFailToast(LinearLayout toastLayout,TextView toastMessage,String message){

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.button_alert_normal));
    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context,toastLayout);
}

//warning toast message method
private final void createWarningToast( LinearLayout toastLayout, TextView toastMessage,
String message) {
toastLayout.setBackgroundColor(context.getResources().getColor(R.color.warning_toast));
    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context, toastLayout);
}

//success toast message method
private final void createSuccessToast(LinearLayout toastLayout,TextView toastMessage,String message){
toastLayout.setBackgroundColor(context.getResources().getColor(R.color.success_toast));

    toastMessage.setText(message);
toastMessage.setTextColor(context.getResources().getColor(R.color.white));
    showToast(context,toastLayout);
}

private void showToast(View view){
Toast toast = new Toast(context);
    toast.setGravity(Gravity.TOP,0,0); // show message in the top of the device
    toast.setDuration(Toast.LENGTH_SHORT);
toast.setView(view);
    toast.show();
}
}

```

第80章：接口

第80.1节：自定义监听器

定义接口

```
//在此接口中，您可以定义将发送给所有者的消息。  
public interface MyCustomListener {  
    //在这种情况下，我们有两个消息，  
    //第一个是在过程成功时发送的消息。  
    void onSuccess(List<Bitmap> bitmapList);  
    //第二个消息是在过程失败时发送的消息。  
    void onFailure(String error);  
}
```

创建监听器

下一步，我们需要在将通过MyCustomListener发送回调的对象中定义一个实例变量。
并为我们的监听器添加设置方法。

```
public class SampleClassB {  
    private MyCustomListener listener;  
  
    public void setMyCustomListener(MyCustomListener listener) {  
        this.listener = listener;  
    }  
}
```

实现监听器

现在，在另一个类中，我们可以创建SampleClassB的实例。

```
public class SomeActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        SampleClassB sampleClass = new SampleClassB();  
    }  
}
```

接下来我们可以通过两种方式将监听器设置给 sampleClass：

通过在我们的类中实现 MyCustomListener：

```
public class SomeActivity extends Activity implements MyCustomListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        SampleClassB sampleClass = new SampleClassB();  
        sampleClass.setMyCustomListener(this);  
    }  
  
    @Override  
    public void onSuccess(List<Bitmap> bitmapList) {  
    }  
  
    @Override  
    public void onFailure(String error) {  
    }  
}
```

Chapter 80: Interfaces

Section 80.1: Custom Listener

Define interface

```
//In this interface, you can define messages, which will be send to owner.  
public interface MyCustomListener {  
    //In this case we have two messages,  
    //the first that is sent when the process is successful.  
    void onSuccess(List<Bitmap> bitmapList);  
    //And The second message, when the process will fail.  
    void onFailure(String error);  
}
```

Create listener

In the next step we need to define an instance variable in the object that will send callback via MyCustomListener.
And add setter for our listener.

```
public class SampleClassB {  
    private MyCustomListener listener;  
  
    public void setMyCustomListener(MyCustomListener listener) {  
        this.listener = listener;  
    }  
}
```

Implement listener

Now, in other class, we can create instance of SampleClassB.

```
public class SomeActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        SampleClassB sampleClass = new SampleClassB();  
    }  
}
```

next we can set our listener, to sampleClass, in two ways:

by implements MyCustomListener in our class:

```
public class SomeActivity extends Activity implements MyCustomListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        SampleClassB sampleClass = new SampleClassB();  
        sampleClass.setMyCustomListener(this);  
    }  
  
    @Override  
    public void onSuccess(List<Bitmap> bitmapList) {  
    }  
  
    @Override  
    public void onFailure(String error) {  
    }  
}
```

```
}
```

或者直接实例化一个匿名内部类：

```
public class SomeActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(new MyCustomListener() {

            @Override
            public void onSuccess(List<Bitmap> bitmapList) {

            }

            @Override
            public void onFailure(String error) {

            }
        });
    }
}
```

触发监听器

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }

    public void doSomething() {
        fetchImages();
    }

    private void fetchImages() {
        AsyncImagefetch imageFetch = new AsyncImageFetch();
        imageFetch.start(new Response<Bitmap>() {
            @Override
            public void onDone(List<Bitmap> bitmapList, Exception e) {
                //如有需要，执行一些操作

                //检查监听器是否已设置。
                if(listener == null)
                    return;
                //触发相应事件。bitmapList 或错误信息将被发送到
                //设置监听器的类。
                if(e == null)
                    listener.onSuccess(bitmapList);
                else
                    listener.onFailure(e.getMessage());
            }
        });
    }
}
```

```
}
```

or just instantiate an anonymous inner class:

```
public class SomeActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(new MyCustomListener() {

            @Override
            public void onSuccess(List<Bitmap> bitmapList) {

            }

            @Override
            public void onFailure(String error) {

            }
        });
    }
}
```

Trigger listener

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }

    public void doSomething() {
        fetchImages();
    }

    private void fetchImages() {
        AsyncImagefetch imageFetch = new AsyncImageFetch();
        imageFetch.start(new Response<Bitmap>() {
            @Override
            public void onDone(List<Bitmap> bitmapList, Exception e) {
                //do some stuff if needed

                //check if listener is set or not.
                if(listener == null)
                    return;
                //Fire proper event. bitmapList or error message will be sent to
                //class which set listener.
                if(e == null)
                    listener.onSuccess(bitmapList);
                else
                    listener.onFailure(e.getMessage());
            }
        });
    }
}
```

第80.2节：基础监听器

“监听器”或“观察者”模式是Android开发中创建异步回调的最常用策略

Section 80.2: Basic Listener

The "listener" or "observer" pattern is the most common strategy for creating asynchronous callbacks in Android

开发。

```
public class MyCustomObject {  
  
    //1 - 定义接口  
    public interface MyCustomObjectListener {  
        public void onAction(String action);  
    }  
  
    //2 - 声明监听器对象  
    private MyCustomObjectListener listener;  
  
    // 并在构造函数中初始化它  
    public MyCustomObject() {  
        this.listener = null;  
    }  
  
    //3 - 创建监听器设置方法  
    public void setCustomObjectListener(MyCustomObjectListener listener) {  
        this.listener = listener;  
    }  
  
    //4 - 触发监听器事件  
    public void makeSomething(){  
        if (this.listener != null){  
            listener.onAction("hello!");  
        }  
    }  
}
```

现在在你的 Activity 中：

```
public class MyActivity extends Activity {  
    public final String TAG = "MyActivity";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main_activity);  
  
        MyCustomObject mObj = new MyCustomObject();  
  
        //5 - 实现监听器回调  
        mObj.setCustomObjectListener(new MyCustomObjectListener() {  
            @Override  
            public void onAction(String action) {  
                Log.d(TAG, "Value: "+action);  
            }  
        });  
    }  
}
```

development.

```
public class MyCustomObject {  
  
    //1 - Define the interface  
    public interface MyCustomObjectListener {  
        public void onAction(String action);  
    }  
  
    //2 - Declare your listener object  
    private MyCustomObjectListener listener;  
  
    // and initialize it in the constructor  
    public MyCustomObject() {  
        this.listener = null;  
    }  
  
    //3 - Create your listener setter  
    public void setCustomObjectListener(MyCustomObjectListener listener) {  
        this.listener = listener;  
    }  
  
    //4 - Trigger listener event  
    public void makeSomething(){  
        if (this.listener != null){  
            listener.onAction("hello!");  
        }  
    }  
}
```

Now on your Activity:

```
public class MyActivity extends Activity {  
    public final String TAG = "MyActivity";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main_activity);  
  
        MyCustomObject mObj = new MyCustomObject();  
  
        //5 - Implement listener callback  
        mObj.setCustomObjectListener(new MyCustomObjectListener() {  
            @Override  
            public void onAction(String action) {  
                Log.d(TAG, "Value: "+action);  
            }  
        });  
    }  
}
```

第81章：动画师

第81.1节：TransitionDrawable动画

此示例显示了一个仅包含两张图片的图像视图的事务。（也可以使用更多图片，依次用于每次事务后的第一层和第二层位置，作为循环）

- 向res/values/arrays.xml 添加图像数组

```
<resources>

<array
    name="splash_images">
    <item>@drawable/splash_img_first</item>
    <item>@drawable/splash_img_second</item>
</array>

</resources>

private Drawable[] backgroundsDrawableArrayForTransition;
private TransitionDrawable transitionDrawable;

private void backgroundAnimTransAction() {

    // 设置资源图片数组
    Resources resources = getResources();
    TypedArray icons = resources.obtainTypedArray(R.array.splash_images);

    @SuppressWarnings(" ResourceType")
    Drawable drawable = icons.getDrawable(0); // 结束图片

    @SuppressWarnings(" ResourceType")
    Drawable drawableTwo = icons.getDrawable(1); // 开始图片

    backgroundsDrawableArrayForTransition = new Drawable[2];
    backgroundsDrawableArrayForTransition[0] = drawable;
    backgroundsDrawableArrayForTransition[1] = drawableTwo;
    transitionDrawable = new TransitionDrawable(backgroundsDrawableArrayForTransition);

    // 你的图片视图 - backgroundImageView
    backgroundImageView.setImageDrawable(transitionDrawable);
    transitionDrawable.startTransition(4000);

    transitionDrawable.setCrossFadeEnabled(false); // 调用公共方法
}

}
```

第81.2节：淡入/淡出动画

为了让视图缓慢淡入或淡出，请使用ObjectAnimator。如下代码所示，使用.setDuration(millis)设置持续时间，其中millis参数是动画的持续时间（以毫秒为单位）。在下面的代码中，视图将在500毫秒（即半秒）内淡入/淡出。要启动ObjectAnimator的动画，调用.start()。动画完成后，会调用 onAnimationEnd(Animator animation)。在这里是设置视图可见性为View.GONE或View.VISIBLE的好地方。

Chapter 81: Animators

Section 81.1: TransitionDrawable animation

This example displays a transaction for an image view with only two images.(can use more images as well one after the other for the first and second layer positions after each transaction as a loop)

- add a image array to res/values/arrays.xml

```
<resources>

<array
    name="splash_images">
    <item>@drawable/splash_img_first</item>
    <item>@drawable/splash_img_second</item>
</array>

</resources>

private Drawable[] backgroundsDrawableArrayForTransition;
private TransitionDrawable transitionDrawable;

private void backgroundAnimTransAction() {

    // set res image array
    Resources resources = getResources();
    TypedArray icons = resources.obtainTypedArray(R.array.splash_images);

    @SuppressWarnings(" ResourceType")
    Drawable drawable = icons.getDrawable(0); // ending image

    @SuppressWarnings(" ResourceType")
    Drawable drawableTwo = icons.getDrawable(1); // starting image

    backgroundsDrawableArrayForTransition = new Drawable[2];
    backgroundsDrawableArrayForTransition[0] = drawable;
    backgroundsDrawableArrayForTransition[1] = drawableTwo;
    transitionDrawable = new TransitionDrawable(backgroundsDrawableArrayForTransition);

    // your image view here - backgroundImageView
    backgroundImageView.setImageDrawable(transitionDrawable);
    transitionDrawable.startTransition(4000);

    transitionDrawable.setCrossFadeEnabled(false); // call public methods
}

}
```

Section 81.2: Fade in/out animation

In order to get a view to slowly fade in or out of view, use an ObjectAnimator. As seen in the code below, set a duration using .setDuration(millis) where the millis parameter is the duration (in milliseconds) of the animation. In the below code, the views will fade in / out over 500 milliseconds, or 1/2 second. To start the ObjectAnimator's animation, call .start(). Once the animation is complete, onAnimationEnd(Animator animation) is called. Here is a good place to set your view's visibility to View.GONE or View.VISIBLE.

```

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ValueAnimator;

void 淡出动画(View 需要淡出的视图) {
    ObjectAnimator 淡出 = ObjectAnimator.ofFloat(需要淡出的视图, "alpha", 1f, 0f);

    淡出.setDuration(500);
    淡出.addListener(new AnimatorListenerAdapter() {
        @Override
        public void 动画结束时(Animator animation) {
            // 我们想在视图淡出后将其设置为GONE, 这样它实际上会从
            // 布局中消失且不占用空间。
            需要淡出的视图.setVisibility(View.GONE);
        }
    });
}

淡出.start();
}

void 淡入动画(View 需要淡入的视图) {
    ObjectAnimator 淡入 = ObjectAnimator.ofFloat(需要淡入的视图, "alpha", 0f, 1f);
    淡入.setDuration(500);

    fadeIn.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationStart(Animator animation) {
            // 我们想将视图设置为可见, 但透明度为0。因此它在布局中看起来是不可见的。
            viewToFadeIn.setVisibility(View.VISIBLE);
            viewToFadeIn.setAlpha(0);
        }
    });
}

fadeIn.start();
}

```

第81.3节：ValueAnimator

ValueAnimator 引入了一种简单的方法来动画化一个值（特定类型，例如 int、float 等）。

通常的使用方法是：

1. 创建一个 ValueAnimator 来动画化一个从 min 到 max 的值
2. 添加一个 UpdateListener，在其中使用计算出的动画值（你可以通过 `getAnimatedValue()` 获取该值）

创建ValueAnimator有两种方法：

(示例代码在250毫秒内将一个float从20f动画到40f)

1. 从xml (放在/res/animator/目录下) :

```

<animator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:valueFrom="20"
    android:valueTo="40"
    android:valueType="floatType"/>

```

ValueAnimator animator = (ValueAnimator) AnimatorInflater.loadAnimator(context,

```

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ValueAnimator;

void fadeOutAnimation(View viewToFadeOut) {
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(viewToFadeOut, "alpha", 1f, 0f);

    fadeOut.setDuration(500);
    fadeOut.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            // We wanna set the view to GONE, after it's fade out. so it actually disappear from the
            // layout & don't take up space.
            viewToFadeOut.setVisibility(View.GONE);
        }
    });
}

fadeOut.start();

void fadeInAnimation(View viewToFadeIn) {
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(viewToFadeIn, "alpha", 0f, 1f);
    fadeIn.setDuration(500);

    fadeIn.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationStart(Animator animation) {
            // We wanna set the view to VISIBLE, but with alpha 0. So it appear invisible in the
            // layout.
            viewToFadeIn.setVisibility(View.VISIBLE);
            viewToFadeIn.setAlpha(0);
        }
    });
}

fadeIn.start();
}

```

Section 81.3: ValueAnimator

ValueAnimator introduces a simple way to animate a value (of a particular type, e.g. `int`, `float`, etc.).

The usual way of using it is:

1. Create a ValueAnimator that will animate a value from min to max
2. Add an `UpdateListener` in which you will use the calculated animated value (which you can obtain with `getAnimatedValue()`)

There are two ways you can create the ValueAnimator:

(the example code animates a `float` from 20f to 40f in 250ms)

1. From xml (put it in the /res/animator/):

```

<animator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:valueFrom="20"
    android:valueTo="40"
    android:valueType="floatType"/>

```

ValueAnimator animator = (ValueAnimator) AnimatorInflater.loadAnimator(context,

```

R.animator.example_animator);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // ... 使用 anim.getAnimatedValue()
    }
});
// 设置所有其他与动画相关的内容 (插值器等)
animator.start();

```

2. 从代码：

```

ValueAnimator animator = ValueAnimator.ofFloat(20f, 40f);
animator.setDuration(250);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // 使用 anim.getAnimatedValue()
    }
});
// 设置所有其他与动画相关的内容 (插值器等)
animator.start();

```

第81.4节：视图的展开和折叠动画

```

public class ViewAnimationUtils {

    public static void expand(final View v) {
        v.measureLayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT);
        final int targetHeight = v.getMeasuredHeight();

        v.getLayoutParams().height = 0;
        v.setVisibility(View.VISIBLE);
        Animation a = new Animation()
        {
            @Override
            protected void applyTransformation(float interpolatedTime, Transformation t) {
                v.getLayoutParams().height = interpolatedTime == 1
                    ? LayoutParams.WRAP_CONTENT
                    : (int)(targetHeight * interpolatedTime);
                v.requestLayout();
            }

            @Override
            public boolean willChangeBounds() {
                return true;
            }
        };

        a.setDuration((int)(targetHeight /
        v.getContext().getResources().getDisplayMetrics().density));
        v.startAnimation(a);
    }

    public static void collapse(final View v) {
        final int initialHeight = v.getMeasuredHeight();

        Animation a = new Animation()
        {
            @Override
            protected void applyTransformation(float interpolatedTime, Transformation t) {

```

```

R.animator.example_animator);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // ... use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();

```

2. From the code:

```

ValueAnimator animator = ValueAnimator.ofFloat(20f, 40f);
animator.setDuration(250);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();

```

Section 81.4: Expand and Collapse animation of View

```

public class ViewAnimationUtils {

    public static void expand(final View v) {
        v.measureLayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT);
        final int targetHeight = v.getMeasuredHeight();

        v.getLayoutParams().height = 0;
        v.setVisibility(View.VISIBLE);
        Animation a = new Animation()
        {
            @Override
            protected void applyTransformation(float interpolatedTime, Transformation t) {
                v.getLayoutParams().height = interpolatedTime == 1
                    ? LayoutParams.WRAP_CONTENT
                    : (int)(targetHeight * interpolatedTime);
                v.requestLayout();
            }

            @Override
            public boolean willChangeBounds() {
                return true;
            }
        };

        a.setDuration((int)(targetHeight /
        v.getContext().getResources().getDisplayMetrics().density));
        v.startAnimation(a);
    }

    public static void collapse(final View v) {
        final int initialHeight = v.getMeasuredHeight();

        Animation a = new Animation()
        {
            @Override
            protected void applyTransformation(float interpolatedTime, Transformation t) {

```

```

        if(interpolatedTime == 1){
            v.setVisibility(View.GONE);
        }else{
            v.setLayoutParams().height = initialHeight - (int)(initialHeight *
            interpolatedTime);
            v.requestLayout();
        }
    }

    @Override
    public boolean willChangeBounds() {
        return true;
    }
};

a.setDuration((int)(initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
v.startAnimation(a);
}
}

```

第81.5节：ObjectAnimator

ObjectAnimator 是 ValueAnimator 的子类，增加了将计算值设置到目标 View 属性的功能。

就像在ValueAnimator中一样，创建ObjectAnimator有两种方式：

(示例代码将一个View的alpha从0.4f动画到0.2f，持续时间为250ms)

1. 从xml (放在/res/animator目录下)

```

<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:propertyName="alpha"
    android:valueFrom="0.4"
    android:valueTo="0.2"
    android:valueType="floatType"/>

```

```

ObjectAnimator animator = (ObjectAnimator) AnimatorInflater.loadAnimator(context,
R.animator.example_animator);
animator.setTarget(exampleView);
// 设置所有你想要的动画相关内容 (插值器等)
animator.start();

```

2. 从代码：

```

ObjectAnimator animator = ObjectAnimator.ofFloat(exampleView, View.ALPHA, 0.4f, 0.2f);
animator.setDuration(250);
// 设置所有你想要的动画相关内容 (插值器等)
animator.start();

```

第81.6节：ViewPropertyAnimator

ViewPropertyAnimator 是一种简化且优化的方式，用于对View的属性进行动画处理。

每个View都可以通过animate()方法获得一个ViewPropertyAnimator对象。你可以使用它通过一次简单的调用同时动画多个属性。每个ViewPropertyAnimator的方法都指定了ViewPropertyAnimator应动画到的特定参数的目标值。

```

        if(interpolatedTime == 1){
            v.setVisibility(View.GONE);
        }else{
            v.setLayoutParams().height = initialHeight - (int)(initialHeight *
            interpolatedTime);
            v.requestLayout();
        }
    }

    @Override
    public boolean willChangeBounds() {
        return true;
    }
};

a.setDuration((int)(initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
v.startAnimation(a);
}
}

```

Section 81.5: ObjectAnimator

ObjectAnimator is a subclass of ValueAnimator with the added ability to set the calculated value to the property of a target View.

Just like in the ValueAnimator, there are two ways you can create the ObjectAnimator:

(the example code animates an alpha of a View from 0.4f to 0.2f in 250ms)

1. From xml (put it in the /res/animator)

```

<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250"
    android:propertyName="alpha"
    android:valueFrom="0.4"
    android:valueTo="0.2"
    android:valueType="floatType"/>

```

```

ObjectAnimator animator = (ObjectAnimator) AnimatorInflater.loadAnimator(context,
R.animator.example_animator);
animator.setTarget(exampleView);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();

```

2. From code:

```

ObjectAnimator animator = ObjectAnimator.ofFloat(exampleView, View.ALPHA, 0.4f, 0.2f);
animator.setDuration(250);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();

```

Section 81.6: ViewPropertyAnimator

ViewPropertyAnimator is a simplified and optimized way to animate properties of a View.

Every single View has a ViewPropertyAnimator object available through the animate() method. You can use that to animate multiple properties at once with a simple call. Every single method of a ViewPropertyAnimator specifies the **target** value of a specific parameter that the ViewPropertyAnimator should animate to.

```

View exampleView = ...;
exampleView.animate()
    .alpha(0.6f)
    .translationY(200)
    .translationXBy(10)
    .scaleX(1.5f)
.setDuration(250)
.setInterpolator(new FastOutLinearInInterpolator());

```

注意：对ViewPropertyAnimator对象调用start()并非必须。如果你不调用，平台会在适当的时间（下一次动画处理周期）自动启动动画。如果你确实调用了start()，则确保动画立即开始。

第81.7节：ImageView的抖动动画

在 res 文件夹下，创建一个名为 "anim" 的新文件夹来存放你的动画资源，并将此文件放入该文件夹中。

shakeanimation.xml

```

<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="100"
    android:fromDegrees="-15"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toDegrees="15" />

```

创建一个名为 Landing 的空白活动

activity_landing.xml

```

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imgBell"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@mipmap/ic_notifications_white_48dp" />

</RelativeLayout>

```

以及在 Landing.java 中为 ImageView 添加动画的方法

Context mContext;

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mContext=this;
    setContentView(R.layout.activity_landing);

    AnimateBell();
}

```

```

View exampleView = ...;
exampleView.animate()
    .alpha(0.6f)
    .translationY(200)
    .translationXBy(10)
    .scaleX(1.5f)
.setDuration(250)
.setInterpolator(new FastOutLinearInInterpolator());

```

Note: Calling start() on a ViewPropertyAnimator object is **NOT** mandatory. If you don't do that you're just letting the platform to handle the starting of the animation in the appropriate time (next animation handling pass). If you actually do that (call start()) you're making sure the animation is started immediately.

Section 81.7: Shake animation of an ImageView

Under res folder, create a new folder called "anim" to store your animation resources and put this on that folder.

shakeanimation.xml

```

<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="100"
    android:fromDegrees="-15"
    android:pivotX="50%"
    android:pivotY="50%"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    android:toDegrees="15" />

```

Create a blank activity called Landing

activity_landing.xml

```

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imgBell"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@mipmap/ic_notifications_white_48dp" />

</RelativeLayout>

```

And the method for animate the imageview on Landing.java

```

Context mContext;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mContext=this;
    setContentView(R.layout.activity_landing);

    AnimateBell();
}

```

```
}

public void AnimateBell() {
    Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
    ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
    imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
    imgBell.setAnimation(shake);
}
```

```
}

public void AnimateBell() {
    Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
    ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
    imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
    imgBell.setAnimation(shake);
}
```

第82章：位置

Android位置API被广泛应用于各种应用中，用于不同目的，例如查找用户位置，在用户离开某个区域时通知（地理围栏），以及帮助识别用户活动（步行、跑步、驾驶等）。

然而，Android 位置 API 并不是获取用户位置的唯一方式。以下将举例说明如何使用 Android 的LocationManager以及其他常用的位置库。

第82.1节：融合位置API

使用带LocationRequest的Activity示例

```
/*
 * 如果您只想在此
 * 活动中接收更新，并且在其他地方不需要使用位置，则此示例非常有用。
 */
public class LocationActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();

        mLocationRequest = new LocationRequest()
            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) // GPS 高精度定位点
            .setInterval(2000) // 至少每 2 秒一次
            .setFastestInterval(1000); // 最多每秒一次
    }

    @Override
    protected void onStart() {
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume() {
        super.onResume();
        // Android 6.0 及以上的权限检查
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
                mLocationRequest, this);
            }
        }
    }
}
```

Chapter 82: Location

Android Location APIs are used in a wide variety of apps for different purposes such as finding user location, notifying when a user has left a general area (Geofencing), and help interpret user activity (walking, running, driving, etc).

However, Android Location APIs are not the only means of acquiring user location. The following will give examples of how to use Android's LocationManager and other common location libraries.

Section 82.1: Fused location API

Example Using Activity w/ LocationRequest

```
/*
 * This example is useful if you only want to receive updates in this
 * activity only, and have no use for location anywhere else.
 */
public class LocationActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();

        mLocationRequest = new LocationRequest()
            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location points
            .setInterval(2000) //At least once every 2 seconds
            .setFastestInterval(1000); //At most once a second
    }

    @Override
    protected void onStart() {
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume() {
        super.onResume();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
                mLocationRequest, this);
            }
        }
    }
}
```

```

@Override
protected void onPause(){
    super.onPause();
    // Android 6.0 及以上的权限检查
    if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        if(mGoogleApiClient.isConnected()) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }
}

@Override
protected void onStop(){
    super.onStop();
    mGoogleApiClient.disconnect();
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
    }
}

@Override
public void onConnectionSuspended(int i) {
    mGoogleApiClient.connect();
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
}

@Override
public void onLocationChanged(Location location) {
    // 在此处理您的位置更新代码
}
}

```

使用带有 PendingIntent 和 BroadcastReceiver 的服务示例

ExampleActivity

推荐阅读：[LocalBroadcastManager](#)

```

/*
 * 如果您有许多不同的类需要接收位置更新，但想对哪些类监听更新进行更细粒度的控制， * 这个
示例非常有用。
 *
 * 例如，当该活动不可见时，它将停止接收更新，但具有注册本地接收器的数据库 * 类将继续接收更新，直到这里调用"stop
Updates()"。
 *
 */
public class ExampleActivity extends AppCompatActivity {

    private InternalLocationReceiver mInternalLocationReceiver;

```

```

@Override
protected void onPause(){
    super.onPause();
    //Permission check for Android 6.0+
    if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        if(mGoogleApiClient.isConnected()) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }
}

@Override
protected void onStop(){
    super.onStop();
    mGoogleApiClient.disconnect();
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
    }
}

@Override
public void onConnectionSuspended(int i) {
    mGoogleApiClient.connect();
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
}

@Override
public void onLocationChanged(Location location) {
    //Handle your location update code here
}
}

```

Example Using Service w/ PendingIntent and BroadcastReceiver

ExampleActivity

Recommended reading: [LocalBroadcastManager](#)

```

/*
 * This example is useful if you have many different classes that should be
 * receiving location updates, but want more granular control over which ones
 * listen to the updates.
 *
 * For example, this activity will stop getting updates when it is not visible, but a database
 * class with a registered local receiver will continue to receive updates, until "stopUpdates()" is
 * called here.
 *
 */
public class ExampleActivity extends AppCompatActivity {

    private InternalLocationReceiver mInternalLocationReceiver;

```

```

@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);

    //仅在此方法中创建内部接收器对象。
    mInternalLocationReceiver = new InternalLocationReceiver(this);
}

@Override
protected void onResume(){
    super.onResume();

    //仅在活动可见时注册以接收更新
    LocalBroadcastManager.getInstance(this).registerReceiver(mInternalLocationReceiver, new
IntentFilter("googleLocation"));
}

@Override
protected void onPause(){
    super.onPause();

    //在活动不可见时注销以停止接收更新。
    //注意：即使此活动被杀死，您仍将收到更新。
    LocalBroadcastManager.getInstance(this).unregisterReceiver(mInternalLocationReceiver);
}

//获取更新的辅助方法
private void requestUpdates(){
startService(new Intent(this, LocationService.class).putExtra("request", true));
}

//辅助方法，用于停止更新
private void stopUpdates(){
startService(new Intent(this, LocationService.class).putExtra("remove", true));
}

/*
 * 用于获取此活动位置更新的内部接收器。
 *
 * 此接收器以及任何通过LocalBroadcastManager注册的接收器
 * 无需在Manifest中注册。
 */
private static class InternalLocationReceiver extends BroadcastReceiver{

    private ExampleActivity mActivity;

    InternalLocationReceiver(ExampleActivity activity){
        mActivity = activity;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        final ExampleActivity activity = mActivity;
        if(activity != null) {
            LocationResult result = intent.getParcelableExtra("result");
            //在此处理位置更新
        }
    }
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);

    //Create internal receiver object in this method only.
    mInternalLocationReceiver = new InternalLocationReceiver(this);
}

@Override
protected void onResume(){
    super.onResume();

    //Register to receive updates in activity only when activity is visible
    LocalBroadcastManager.getInstance(this).registerReceiver(mInternalLocationReceiver, new
IntentFilter("googleLocation"));
}

@Override
protected void onPause(){
    super.onPause();

    //Unregister to stop receiving updates in activity when it is not visible.
    //NOTE: You will still receive updates even if this activity is killed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(mInternalLocationReceiver);
}

//Helper method to get updates
private void requestUpdates(){
    startService(new Intent(this, LocationService.class).putExtra("request", true));
}

//Helper method to stop updates
private void stopUpdates(){
    startService(new Intent(this, LocationService.class).putExtra("remove", true));
}

/*
 * Internal receiver used to get location updates for this activity.
 *
 * This receiver and any receiver registered with LocalBroadcastManager does
 * not need to be registered in the Manifest.
 */
private static class InternalLocationReceiver extends BroadcastReceiver{

    private ExampleActivity mActivity;

    InternalLocationReceiver(ExampleActivity activity){
        mActivity = activity;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        final ExampleActivity activity = mActivity;
        if(activity != null) {
            LocationResult result = intent.getParcelableExtra("result");
            //Handle location update here
        }
    }
}

```

LocationService

注意：别忘了在清单文件中注册此服务！

```
public class LocationService extends Service implements
    GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    public void onCreate(){
        super.onCreate();
    }
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
    .build();

    mLocationRequest = new LocationRequest()
        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) // GPS 高精度定位点
        .setInterval(2000) // 至少每 2 秒一次
    .setFastestInterval(1000); // 最多每秒一次
}

@Override
public int onStartCommand(Intent intent, int flags, int startId){
    super.onStartCommand(intent, flags, startId);
    // Android 6.0 及以上的权限检查
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        if (intent.getBooleanExtra("request", false)) {
            if (mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
            } else {
                mGoogleApiClient.connect();
            }
        } else if(intent.getBooleanExtra("remove", false)){
            stopSelf();
        }
    }
    return START_STICKY;
}

@Override
public void onDestroy(){
    super.onDestroy();
    if(mGoogleApiClient.isConnected()){
        LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
getPendingIntent());
        mGoogleApiClient.disconnect();
    }
}

private PendingIntent getPendingIntent(){

    //IntentService 示例
    //return PendingIntent.getService(this, 0, new Intent(this,
    **YOUR_INTENT_SERVICE_CLASS_HERE**), PendingIntent.FLAG_UPDATE_CURRENT);
}
```

LocationService

NOTE: Don't forget to register this service in the Manifest!

```
public class LocationService extends Service implements
    GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    public void onCreate(){
        super.onCreate();
        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
        .build();

        mLocationRequest = new LocationRequest()
            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location points
            .setInterval(2000) //At least once every 2 seconds
        .setFastestInterval(1000); //At most once a second
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        super.onStartCommand(intent, flags, startId);
        //Permission check for Android 6.0+
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if (intent.getBooleanExtra("request", false)) {
                if (mGoogleApiClient.isConnected()) {
                    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
                } else {
                    mGoogleApiClient.connect();
                }
            } else if(intent.getBooleanExtra("remove", false)){
                stopSelf();
            }
        }
        return START_STICKY;
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        if(mGoogleApiClient.isConnected()){
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
getPendingIntent());
            mGoogleApiClient.disconnect();
        }
    }

    private PendingIntent getPendingIntent(){

        //Example for IntentService
        //return PendingIntent.getService(this, 0, new Intent(this,
        **YOUR_INTENT_SERVICE_CLASS_HERE**), PendingIntent.FLAG_UPDATE_CURRENT);
    }
}
```

```

//BroadcastReceiver 例
return PendingIntent.getBroadcast(this, 0, new Intent(this, LocationReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT);
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    //Android 6.0+ 的权限检查
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
    }
}

@Override
public void onConnectionSuspended(int i) {
    mGoogleApiClient.connect();
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

位置接收器

注意：别忘了在清单文件中注册此接收器！

```

public class 位置接收器 extends 广播接收器 {

    @Override
    public void onReceive(上下文 context, Intent intent) {
        if(位置结果.hasResult(intent)){
            位置结果 locationResult = 位置结果.extractResult(intent);
            本地广播管理器.getInstance(context).sendBroadcast(new
Intent("googleLocation").putExtra("result", locationResult));
        }
    }
}

```

第82.2节：使用地理编码器从位置获取地址

在从融合API获取到位置对象后，你可以轻松地从该对象获取地址信息。

```

private Address getCountryInfo(Location location) {
    Address address = null;
    Geocoder 地理编码器 = new Geocoder(getActivity(), Locale.getDefault());
    String 错误信息;
    List<Address> 地址列表 = null;
    try {
        地址列表 = 地理编码器.getFromLocation(

```

```

//Example for BroadcastReceiver
return PendingIntent.getBroadcast(this, 0, new Intent(this, LocationReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT);
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    //Permission check for Android 6.0+
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
    }
}

@Override
public void onConnectionSuspended(int i) {
    mGoogleApiClient.connect();
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

LocationReceiver

NOTE: Don't forget to register this receiver in the Manifest!

```

public class LocationReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(LocationResult.hasResult(intent)){
            LocationResult locationResult = LocationResult.extractResult(intent);
            LocalBroadcastManager.getInstance(context).sendBroadcast(new
Intent("googleLocation").putExtra("result", locationResult));
        }
    }
}

```

Section 82.2: Get Address From Location using Geocoder

After you got the Location object from FusedAPI, you can easily acquire Address information from that object.

```

private Address getCountryInfo(Location location) {
    Address address = null;
    Geocoder geocoder = new Geocoder(getActivity(), Locale.getDefault());
    String errorMessage;
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(

```

```

位置.getLatitude(),
位置.getLongitude(),
// 在此示例中，仅获取单个地址。
1);
} catch (IOException ioException) {
// 捕获网络或其他输入输出异常。
错误信息 = "IOException>>" + ioException.getMessage();
} catch (IllegalArgumentException illegalArgumentException) {
// 捕获无效的纬度或经度值。
errorMessage = "IllegalArgumentException>>" + illegalArgumentException.getMessage();
}
if (addresses != null && !addresses.isEmpty()) {
address = addresses.get(0);
}
return country;
}

```

第82.3节：使用 LocationManager请求位置更新

一如既往，您需要确保拥有所需的权限。

```

public class MainActivity extends AppCompatActivity implements LocationListener{

private LocationManager mLocationManager = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main2);

mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
}

@Override
protected void onResume() {
super.onResume();

try {
mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
} catch(SecurityException e){
// 应用程序没有正确的权限
}
}

@Override
protected void onPause() {
try{
mLocationManager.removeUpdates(this);
} catch (SecurityException e){
// 应用程序没有正确的权限
}

super.onPause();
}
}

```

```

location.getLatitude(),
location.getLongitude(),
// In this sample, get just a single address.
1);
} catch (IOException ioException) {
// Catch network or other I/O problems.
errorMessage = "IOException>>" + ioException.getMessage();
} catch (IllegalArgumentException illegalArgumentException) {
// Catch invalid latitude or longitude values.
errorMessage = "IllegalArgumentException>>" + illegalArgumentException.getMessage();
}
if (addresses != null && !addresses.isEmpty()) {
address = addresses.get(0);
}
return country;
}

```

Section 82.3: Requesting location updates using LocationManager

As always, you need to make sure you have the required permissions.

```

public class MainActivity extends AppCompatActivity implements LocationListener{

private LocationManager mLocationManager = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main2);

mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
}

@Override
protected void onResume() {
super.onResume();

try {
mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
} catch(SecurityException e){
// The app doesn't have the correct permissions
}

}

@Override
protected void onPause() {
try{
mLocationManager.removeUpdates(this);
} catch (SecurityException e){
// The app doesn't have the correct permissions
}

super.onPause();
}
}

```

```

@Override
public void onLocationChanged(Location location) {
    // 我们收到了位置更新!
Log.i("onLocationChanged", location.toString());
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
}

@Override
public void onProviderEnabled(String provider) {
}

@Override
public void onProviderDisabled(String provider) {
}
}

```

```

@Override
public void onLocationChanged(Location location) {
    // We received a location update!
    Log.i("onLocationChanged", location.toString());
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
}

@Override
public void onProviderEnabled(String provider) {
}

@Override
public void onProviderDisabled(String provider) {
}
}

```

第82.4节：使用LocationManager在单独线程上请求位置更新

一如既往，您需要确保拥有所需的权限。

```

public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;
    HandlerThread mLocationHandlerThread = null;
    Looper mLocationHandlerLooper = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        mLocationHandlerThread = new HandlerThread("locationHandlerThread");
    }

    @Override
    protected void onResume() {
        super.onResume();

        mLocationHandlerThread.start();
        mLocationHandlerLooper = mLocationHandlerThread.getLooper();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this,
            mLocationHandlerLooper);
        }
        catch(SecurityException e){
            // 应用程序没有正确的权限
        }
    }
}

```

Section 82.4: Requesting location updates on a separate thread using LocationManager

As always, you need to make sure you have the required permissions.

```

public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;
    HandlerThread mLocationHandlerThread = null;
    Looper mLocationHandlerLooper = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        mLocationHandlerThread = new HandlerThread("locationHandlerThread");
    }

    @Override
    protected void onResume() {
        super.onResume();

        mLocationHandlerThread.start();
        mLocationHandlerLooper = mLocationHandlerThread.getLooper();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this,
            mLocationHandlerLooper);
        }
        catch(SecurityException e){
            // The app doesn't have the correct permissions
        }
    }
}

```

```

@Override
protected void onPause() {
    try{
mLocationManager.removeUpdates(this);
    }
    catch (SecurityException e){
        // 应用程序没有正确的权限
    }

mLocationHandlerLooper = null;

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2)
        mLocationHandlerThread.quitSafely();
    否则
mLocationHandlerThread.quit();

    mLocationHandlerThread = null;

    super.onPause();
}
}

@Override
public void onLocationChanged(Location location) {
    // 我们在一个单独的线程上收到了位置更新！
Log.i("onLocationChanged", location.toString());

    // 你可以通过如下方式验证当前线程：
    // Log.d("当前线程?", Thread.currentThread() == Looper.getMainLooper().getThread() ? "UI
线程" : "新线程");
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {

}

@Override
public void onProviderEnabled(String provider) {

}

@Override
public void onProviderDisabled(String provider) {
}
}

```

第82.5节：在BroadcastReceiver中获取位置更新

首先创建一个BroadcastReceiver类来处理传入的位置更新：

```

public class LocationReceiver extends BroadcastReceiver implements Constants {

@Override
public void onReceive(Context context, Intent intent) {

    if (LocationResult.hasResult(intent)) {

```

```

@Override
protected void onPause() {
    try{
mLocationManager.removeUpdates(this);
    }
    catch (SecurityException e){
        // The app doesn't have the correct permissions
    }

mLocationHandlerLooper = null;

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2)
        mLocationHandlerThread.quitSafely();
    else
        mLocationHandlerThread.quit();

    mLocationHandlerThread = null;

    super.onPause();
}
}

@Override
public void onLocationChanged(Location location) {
    // We received a location update on a separate thread!
Log.i("onLocationChanged", location.toString());

    // You can verify which thread you're on by something like this:
    // Log.d("Which thread?", Thread.currentThread() == Looper.getMainLooper().getThread() ? "UI
Thread" : "New thread");
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {

}

@Override
public void onProviderEnabled(String provider) {

}

@Override
public void onProviderDisabled(String provider) {
}
}

```

Section 82.5: Getting location updates in a BroadcastReceiver

First create a BroadcastReceiver class to handle the incoming Location updates:

```

public class LocationReceiver extends BroadcastReceiver implements Constants {

@Override
public void onReceive(Context context, Intent intent) {

    if (LocationResult.hasResult(intent)) {

```

```

LocationResult locationResult = LocationResult.extractResult(intent);
    Location location = locationResult.getLastLocation();
    if (location != null) {
        // 处理您的位置
    } else {
        Log.d(LocationReceiver.class.getSimpleName(), "*** 位置对象为空 ***");
    }
}
}
}

```

然后当您在 `onConnected` 回调中连接到 `GoogleApiClient` 时：

```

@Override
public void onConnected(Bundle connectionHint) {
    Intent backgroundIntent = new Intent(this, LocationReceiver.class);
    mBackgroundPendingIntent = backgroundIntent.getBroadcast(getApplicationContext(),
LOCATION_REQUEST_CODE, backgroundIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    mFusedLocationProviderApi.requestLocationUpdates(mLocationClient, mLocationRequest,
backgroundPendingIntent);
}

```

别忘了在适当的生命周期回调中移除位置更新的 intent：

```

@Override
public void onDestroy() {
    if (servicesAvailable && mLocationClient != null) {
        if (mLocationClient.isConnected()) {
            fusedLocationProviderApi.removeLocationUpdates(mLocationClient,
backgroundPendingIntent);
            // 销毁当前位置客户端
        }
        mLocationClient = null;
    } 否则 {
        mLocationClient.unregisterConnectionCallbacks(this);
        mLocationClient = null;
    }
    super.onDestroy();
}

```

第82.6节：注册地理围栏

我已创建了 `GeoFenceObservationService` 单例类。

`GeoFenceObservationService.java`:

```

public class GeoFenceObservationService extends Service implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
ResultCallback<Status> {

protected static final String TAG = "GeoFenceObservationService";
protected GoogleApiClient mGoogleApiClient;
protected ArrayList<Geofence> mGeofenceList;
private boolean mGeofencesAdded;
private SharedPreferences mSharedPreferences;
private static GeoFenceObservationService mInstant;
public static GeoFenceObservationService getInstant(){
    return mInstant;
}
}

```

```

LocationResult locationResult = LocationResult.extractResult(intent);
Location location = locationResult.getLastLocation();
if (location != null) {
    // Do something with your location
} else {
    Log.d(LocationReceiver.class.getSimpleName(), "*** location object is null ***");
}
}
}
}

```

Then when you connect to the `GoogleApiClient` in the `onConnected` callback:

```

@Override
public void onConnected(Bundle connectionHint) {
    Intent backgroundIntent = new Intent(this, LocationReceiver.class);
    mBackgroundPendingIntent = backgroundIntent.getBroadcast(getApplicationContext(),
LOCATION_REQUEST_CODE, backgroundIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    mFusedLocationProviderApi.requestLocationUpdates(mLocationClient, mLocationRequest,
backgroundPendingIntent);
}

```

Don't forget to remove the location update intent in the appropriate lifecycle callback:

```

@Override
public void onDestroy() {
    if (servicesAvailable && mLocationClient != null) {
        if (mLocationClient.isConnected()) {
            fusedLocationProviderApi.removeLocationUpdates(mLocationClient,
backgroundPendingIntent);
            // Destroy the current location client
            mLocationClient = null;
        } else {
            mLocationClient.unregisterConnectionCallbacks(this);
            mLocationClient = null;
        }
    }
    super.onDestroy();
}

```

Section 82.6: Register geofence

I have created `GeoFenceObservationService` **singleton** class.

`GeoFenceObservationService.java`:

```

public class GeoFenceObservationService extends Service implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
ResultCallback<Status> {

protected static final String TAG = "GeoFenceObservationService";
protected GoogleApiClient mGoogleApiClient;
protected ArrayList<Geofence> mGeofenceList;
private boolean mGeofencesAdded;
private SharedPreferences mSharedPreferences;
private static GeoFenceObservationService mInstant;
public static GeoFenceObservationService getInstant(){
    return mInstant;
}
}

```

```

@Override
public void onCreate() {
    super.onCreate();
mInstant = this;
mGeofenceList = new ArrayList<Geofence>();
    mSharedPreferences = getSharedPreferences(AppConstants.SHARED_PREFERENCES_NAME,
    MODE_PRIVATE);
    mGeofencesAdded = mSharedPreferences.getBoolean(AppConstants.GEOFENCES_ADDED_KEY, false);

    buildGoogleApiClient();
}

@Override
public void onDestroy() {
    mGoogleApiClient.disconnect();
    super.onDestroy();
}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    return START_STICKY;
}

protected void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
    .build();
    mGoogleApiClient.connect();
}

@Override
public void onConnected(Bundle connectionHint) {
}

@Override
public void onConnectionFailed(ConnectionResult result) {
}

@Override
public void onConnectionSuspended(int cause) {
}

private GeofencingRequest getGeofencingRequest() {

GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(mGeofenceList);
    return builder.build();
}

public void addGeofences() {

```

```

@Override
public void onCreate() {
    super.onCreate();
mInstant = this;
mGeofenceList = new ArrayList<Geofence>();
    mSharedPreferences = getSharedPreferences(AppConstants.SHARED_PREFERENCES_NAME,
    MODE_PRIVATE);
    mGeofencesAdded = mSharedPreferences.getBoolean(AppConstants.GEOFENCES_ADDED_KEY, false);

    buildGoogleApiClient();
}

@Override
public void onDestroy() {
    mGoogleApiClient.disconnect();
    super.onDestroy();
}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    return START_STICKY;
}

protected void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}

@Override
public void onConnected(Bundle connectionHint) {
}

@Override
public void onConnectionFailed(ConnectionResult result) {
}

@Override
public void onConnectionSuspended(int cause) {
}

private GeofencingRequest getGeofencingRequest() {

GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(mGeofenceList);
    return builder.build();
}

public void addGeofences() {

```

```

        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
            return;
        }

        populateGeofenceList();
        if(!mGeofenceList.isEmpty()){
            try {
                LocationServices.GeofencingApi.addGeofences(mGoogleApiClient,
                    getGeofencingRequest(), getGeofencePendingIntent()).setResultCallback(this);
            } catch (SecurityException securityException) {
                securityException.printStackTrace();
            }
        }
    }

    public void removeGeofences() {
        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
            return;
        }
        try {

            LocationServices.GeofencingApi.removeGeofences(mGoogleApiClient, getGeofencePendingIntent()).setResultCallback(this);
        } catch (SecurityException securityException) {
            securityException.printStackTrace();
        }
    }

    public void onResult(Status status) {

        if (status.isSuccess()) {
            mGeofencesAdded = !mGeofencesAdded;
            Sharedpreferences.Editor editor = mSharedpreferences.edit();
            editor.putBoolean(AppConstants.GEOFENCES_ADDED_KEY, mGeofencesAdded);
            editor.apply();
        } else {
            String errorMessage = AppConstants.getErrorString(this, status.getStatusCode());
            Log.i("Geofence", errorMessage);
        }
    }

    private PendingIntent getGeofencePendingIntent() {
        Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);
        return PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    }

    private void populateGeofenceList() {
        mGeofenceList.clear();
        List<GeoFencingResponce> geoFenceList = getGeofencesList();
        if(geoFenceList!=null&&!geoFenceList.isEmpty()){
            for (GeoFencingResponce obj : geoFenceList){
                mGeofenceList.add(obj.getGeofence());
                Log.i(TAG, "已注册的地理围栏：" + obj.Id+"-"+obj.Name+"-"+obj.Latitude+"-
"+obj.Longitude);
            }
        }
    }
}

```

```

        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
            return;
        }

        populateGeofenceList();
        if(!mGeofenceList.isEmpty()){
            try {
                LocationServices.GeofencingApi.addGeofences(mGoogleApiClient,
                    getGeofencingRequest(), getGeofencePendingIntent()).setResultCallback(this);
            } catch (SecurityException securityException) {
                securityException.printStackTrace();
            }
        }
    }

    public void removeGeofences() {
        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
            return;
        }
        try {

            LocationServices.GeofencingApi.removeGeofences(mGoogleApiClient, getGeofencePendingIntent()).setResultCallback(this);
        } catch (SecurityException securityException) {
            securityException.printStackTrace();
        }
    }

    public void onResult(Status status) {

        if (status.isSuccess()) {
            mGeofencesAdded = !mGeofencesAdded;
            Sharedpreferences.Editor editor = mSharedpreferences.edit();
            editor.putBoolean(AppConstants.GEOFENCES_ADDED_KEY, mGeofencesAdded);
            editor.apply();
        } else {
            String errorMessage = AppConstants.getErrorString(this, status.getStatusCode());
            Log.i("Geofence", errorMessage);
        }
    }

    private PendingIntent getGeofencePendingIntent() {
        Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);
        return PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    }

    private void populateGeofenceList() {
        mGeofenceList.clear();
        List<GeoFencingResponce> geoFenceList = getGeofencesList();
        if(geoFenceList!=null&&!geoFenceList.isEmpty()){
            for (GeoFencingResponce obj : geoFenceList){
                mGeofenceList.add(obj.getGeofence());
                Log.i(TAG, "Registered Geofences：" + obj.Id+"-"+obj.Name+"-"+obj.Latitude+"-
"+obj.Longitude);
            }
        }
    }
}

```

AppConstant:

```
public static final String SHARED_PREFERENCES_NAME = PACKAGE_NAME + ".SHARED_PREFERENCES_NAME";
public static final String GEOFENCES_ADDED_KEY = PACKAGE_NAME + ".GEOFENCES_ADDED_KEY";
public static final String DETECTED_GEOFENCES = "detected_geofences";
public static final String DETECTED_BEACONS = "detected_beacons";

public static String getErrorString(Context context, int errorCode) {
    Resources mResources = context.getResources();
    switch (errorCode) {
        case GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE:
            return mResources.getString(R.string.geofence_not_available);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_GEOFENCES:
            return mResources.getString(R.string.geofence_too_many_geofences);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_PENDING_INTENTS:
            return mResources.getString(R.string.geofence_too_many_pending_intents);
        default:
            return mResources.getString(R.string.unknown_geofence_error);
    }
}
```

我从哪里启动服务？从 Application 类开始

- startService(new Intent(getApplicationContext(),GeoFenceObservationService.class));

我如何注册地理围栏？

- GeoFenceObservationService.getInstant().addGeofences();

AppConstant:

```
public static final String SHARED_PREFERENCES_NAME = PACKAGE_NAME + ".SHARED_PREFERENCES_NAME";
public static final String GEOFENCES_ADDED_KEY = PACKAGE_NAME + ".GEOFENCES_ADDED_KEY";
public static final String DETECTED_GEOFENCES = "detected_geofences";
public static final String DETECTED_BEACONS = "detected_beacons";

public static String getErrorString(Context context, int errorCode) {
    Resources mResources = context.getResources();
    switch (errorCode) {
        case GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE:
            return mResources.getString(R.string.geofence_not_available);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_GEOFENCES:
            return mResources.getString(R.string.geofence_too_many_geofences);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_PENDING_INTENTS:
            return mResources.getString(R.string.geofence_too_many_pending_intents);
        default:
            return mResources.getString(R.string.unknown_geofence_error);
    }
}
```

Where I started Service ? From Application class

- startService(**new Intent(getApplicationContext(),GeoFenceObservationService.class)**);

How I registered Geofences ?

- GeoFenceObservationService.getInstant().addGeofences();

第83章：主题、样式、属性

第83.1节：定义主色、主暗色和强调色

您可以自定义您的主题的颜色调色板。

使用框架API

版本 ≥ 5.0

```
<style name="AppTheme" parent="Theme.Material">
    <item name="android:colorPrimary">@color/primary</item>
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <item name="android:colorAccent">@color/accent</item>
</style>
```

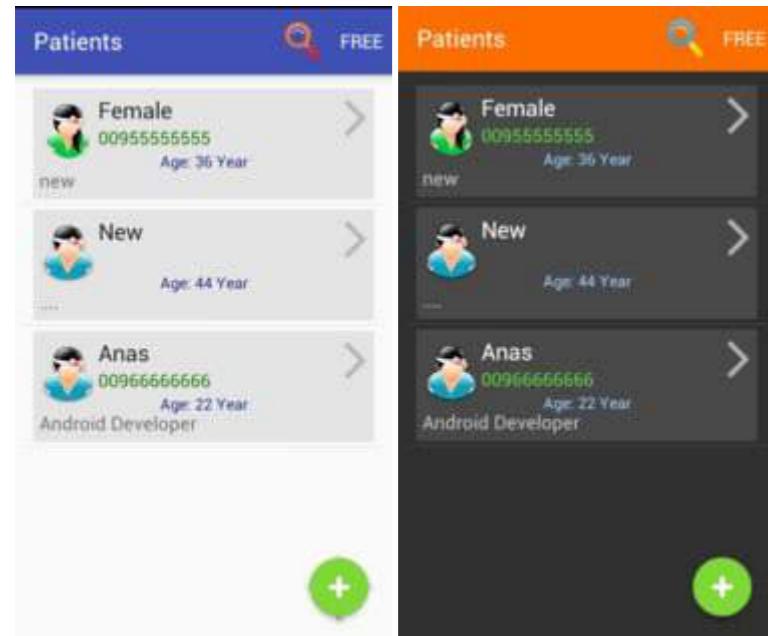
使用Appcompat支持库（和AppCompatActivity）

版本 ≥ 2.1.x

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

第83.2节：一个应用中的多主题

在您的Android应用中使用多个主题，您可以为每个主题添加自定义颜色，如下所示：



首先，我们必须像这样将主题添加到style.xml中：

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">
</style>

<!-- -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >
```

Chapter 83: Theme, Style, Attribute

Section 83.1: Define primary, primary dark, and accent colors

You can customize your [theme's color palette](#).

Using framework APIs

Version ≥ 5.0

```
<style name="AppTheme" parent="Theme.Material">
    <item name="android:colorPrimary">@color/primary</item>
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <item name="android:colorAccent">@color/accent</item>
</style>
```

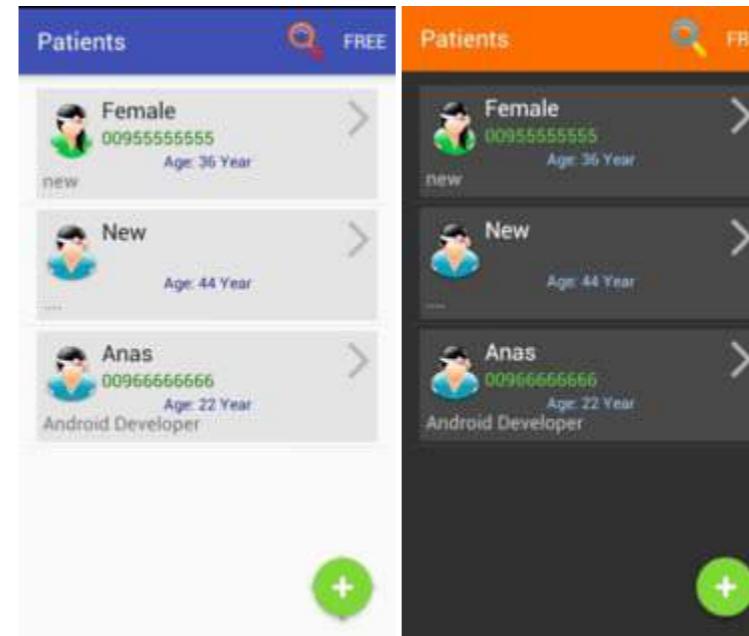
Using the Appcompat support library (and AppCompatActivity)

Version ≥ 2.1.x

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

Section 83.2: Multiple Themes in one App

Using more than one theme in your Android application, you can add custom colors to every theme, to be like this:



First, we have to add our themes to style.xml like this:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">
</style>

<!-- -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >
```

```
</style>
```

.....

上面你可以看到OneTheme和TwoTheme。

现在，打开你的AndroidManifest.xml文件，添加这一行：android:theme="@style/OneTheme"到你的application标签中，这将使OneTheme成为默认主题：

```
<application  
    android:theme="@style/OneTheme"  
    ...>
```

新建一个名为attrs.xml的文件，并添加以下代码：

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <attr name="custom_red" format="color" />  
    <attr name="custom_blue" format="color" />  
    <attr name="custom_green" format="color" />  
</resources>  
<!-- 添加你需要的所有颜色（只需颜色名称） -->
```

返回到style.xml并为每个主题添加这些颜色及其对应的值：

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
    <item name="custom_red">#8b030c</item>  
    <item name="custom_blue">#0f1b8b</item>  
    <item name="custom_green">#1c7806</item>  
</style>  
  
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >  
    <item name="custom_red">#ff606b</item>  
    <item name="custom_blue">#99cff</item>  
    <item name="custom_green">#62e642</item>  
</style>
```

现在你已经为每个主题定义了自定义颜色，接下来将这些颜色添加到我们的视图中。

通过使用"?attr/"，将custom_blue颜色添加到TextView：

进入你的ImageView并添加此颜色：

```
<TextView>  
    android:id="@+id/txt_e_view"  
    android:textColor="?attr/custom_blue" />
```

现在我们只需一行代码就能更改主题setTheme(R.style.TwoTheme)；这行代码必须放在setContentView()方法之前，位于onCreate()方法中，示例如下Activity.java：

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setTheme(R.style.TwoTheme);  
    setContentView(R.layout.main_activity);  
    ...  
}
```

```
</style>
```

.....

Above you can see **OneTheme** and **TwoTheme**.

Now, go to your `AndroidManifest.xml` and add this line: `android:theme="@style/OneTheme"` to your `application` tag, this will make **OneTheme** the default theme:

```
<application  
    android:theme="@style/OneTheme"  
    ...>
```

Create new xml file named `attrs.xml` and add this code :

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <attr name="custom_red" format="color" />  
    <attr name="custom_blue" format="color" />  
    <attr name="custom_green" format="color" />  
</resources>  
<!-- add all colors you need (just color's name) -->
```

Go back to `style.xml` and add these colors with its values for each theme :

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
    <item name="custom_red">#8b030c</item>  
    <item name="custom_blue">#0f1b8b</item>  
    <item name="custom_green">#1c7806</item>  
</style>  
  
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >  
    <item name="custom_red">#ff606b</item>  
    <item name="custom_blue">#99cff</item>  
    <item name="custom_green">#62e642</item>  
</style>
```

Now you have custom colors for each theme, let's add these color to our views.

Add **custom_blue** color to the TextView by using "?attr/" :

Go to your imageView and add this color :

```
<TextView>  
    android:id="@+id/txt_e_view"  
    android:textColor="?attr/custom_blue" />
```

Now we can change the theme just by single line `setTheme(R.style.TwoTheme)` ; this line must be before `setContentView()` method in `onCreate()` method, like this `Activity.java` :

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setTheme(R.style.TwoTheme);  
    setContentView(R.layout.main_activity);  
    ...  
}
```

一次性更改所有活动的主题

如果我们想更改所有活动的主题，需要创建一个名为MyActivity的类，继承自AppCompatActivity类（或Activity类），并在onCreate()方法中添加setTheme(R.style.TwoTheme);这行代码：

```
public class MyActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        if (new MySettings(this).isDarkTheme())  
            setTheme(R.style.TwoTheme);  
    }  
}
```

最后，进入所有活动，将它们全部继承自MyActivity基类：

```
public class MainActivity extends MyActivity {  
    ...  
}
```

要更改主题，只需进入**MyActivity**并将R.style.TwoTheme更改为您的主题（R.style.OneTheme, R.style.ThreeTheme 等）。

第83.3节：导航栏颜色 (API 21及以上)

版本 ≥ 5.0

此属性用于更改导航栏（包含返回、主页、最近按钮的那个）。通常它是黑色的，但颜色可以更改。

```
<style name="AppTheme" parent="Theme.AppCompat">  
    <item name="android:navigationBarColor">@color/my_color</item>  
</style>
```

第83.4节：为每个Activity使用自定义主题

在themes.xml中：

```
<style name="MyActivityTheme" parent="Theme.AppCompat">  
    <!-- 主题属性写在这里 -->  
</style>
```

在AndroidManifest.xml中：

```
<application  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"  
    android:theme="@style/Theme.AppCompat">  
  
    <activity  
        android:name=".MyActivity"  
        android:theme="@style/MyActivityTheme" />  
    </application>
```

change theme for all activities at once

If we want to change the theme for all activities, we have to create new class named MyActivity extends AppCompatActivity class (or Activity class) and add line setTheme(R.style.TwoTheme); to **onCreate()** method:

```
public class MyActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        if (new MySettings(this).isDarkTheme())  
            setTheme(R.style.TwoTheme);  
    }  
}
```

Finally, go to all your activities add make all of them extend the **MyActivity** base class:

```
public class MainActivity extends MyActivity {  
    ...  
}
```

In order to change the theme, just go to **MyActivity** and change R.style.TwoTheme to your theme (R.style.OneTheme, R.style.ThreeTheme).

Section 83.3: Navigation Bar Color (API 21+)

Version ≥ 5.0

This attribute is used to change the navigation bar (one, that contain Back, Home Recent button). Usually it is black, however its color can be changed.

```
<style name="AppTheme" parent="Theme.AppCompat">  
    <item name="android:navigationBarColor">@color/my_color</item>  
</style>
```

Section 83.4: Use Custom Theme Per Activity

In themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">  
    <!-- Theme attributes here -->  
</style>
```

In AndroidManifest.xml:

```
<application  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"  
    android:theme="@style/Theme.AppCompat">  
  
    <activity  
        android:name=".MyActivity"  
        android:theme="@style/MyActivityTheme" />  
    </application>
```

第83.5节：浅色状态栏 (API 23及以上)

此属性可以将状态栏图标（屏幕顶部）的背景更改为白色。

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

第83.6节：全局使用自定义主题

在themes.xml中：

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- 主题属性在此处 -->
</style>
```

在AndroidManifest.xml中：

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- 活动声明在此处 -->

```

第83.7节：过度滚动颜色 (API 21及以上)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

第83.8节：波纹颜色 (API 21及以上)

版本 ≥ 5.0

当用户按下可点击视图时，会显示波纹动画。

您可以通过在视图中分配?android:colorControlHighlight来使用应用程序中相同的波纹颜色。

您可以通过更改主题中的android:colorControlHighlight属性来自定义此颜色：

此效果颜色可以更改：

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

或者，如果您使用的是 Material 主题：

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Section 83.5: Light Status Bar (API 23+)

This attribute can change the background of the Status Bar icons (at the top of the screen) to white.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Section 83.6: Use Custom Theme Globally

In themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->
</application>
```

Section 83.7: Overscroll Color (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Section 83.8: Ripple Color (API 21+)

Version ≥ 5.0

The ripple animation is shown when user presses clickable views.

You can use the same ripple color used by your app assigning the ?android:colorControlHighlight in your views.

You can customize this color by changing the android:colorControlHighlight attribute in your theme:

This effect color can be changed:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Or, if you are using a Material Theme:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

第 83.9 节：半透明导航栏和状态栏 (API 19 及以上)

导航栏（屏幕底部）可以是透明的。以下是实现方法。

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

状态栏（屏幕顶部）可以通过将此属性应用于样式来设置为透明：

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

第83.10节：主题继承

定义主题时，通常使用系统提供的主题，然后修改外观以适应自己的应用。例如，继承Theme.AppCompat主题的方法如下：

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

该主题现在拥有标准Theme.AppCompat主题的所有属性，除了我们明确修改的那些。

继承时还有一个快捷方式，通常用于从自己的主题继承：

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

由于名称开头已经包含AppTheme.，它会自动继承该主题，无需定义parent主题。当你需要为应用的某个部分（例如单个Activity）创建特定样式时，这非常有用。

Section 83.9: Translucent Navigation and Status Bars (API 19+)

The navigation bar (at the bottom of the screen) can be transparent. Here is the way to achieve it.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

The Status Bar (top of the screen) can be made transparent, by applying this attribute to the style:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Section 83.10: Theme inheritance

When defining themes, one usually uses the theme provided by the system, and then changes modifies the look to fit his own application. For example, this is how the Theme.AppCompat theme is inherited:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

This theme now has all the properties of the standard Theme.AppCompat theme, except the ones we explicitly changed.

There is also a shortcut when inheriting, usually used when one inherits from his own theme:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Since it already has AppTheme. in the start of its name, it automatically inherits it, without needing to define the parent theme. This is useful when you need to create specific styles for a part (for example, a single Activity) of your app.

第84章：MediaPlayer

第84.1节：基本创建与播放

MediaPlayer类可用于控制音频/视频文件和流的播放。

MediaPlayer对象的创建有三种类型：

- 来自本地资源的媒体

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // 无需调用prepare()；create()会为你完成
```

- 来自本地URI（通过ContentResolver获取）

```
Uri myUri = ....; // 在此初始化Uri
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

- 来自外部URL

```
String url = "http://....."; // 你的URL地址
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // 可能需要较长时间！（用于缓冲等）
mediaPlayer.start();
```

第84.2节：带缓冲进度和播放位置的媒体播放器

```
public class SoundActivity extends Activity {

    private MediaPlayer mediaPlayer;
    ProgressBar progress_bar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tool_sound);
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        progress_bar = (ProgressBar) findViewById(R.id.progress_bar);

        btn_play_stop.setEnabled(false);
        btn_play_stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mediaPlayer.isPlaying()) {
                    mediaPlayer.pause();
                    btn_play_stop.setImageResource(R.drawable.ic_pause_black_24dp);
                } else {
                    mediaPlayer.start();
                }
            }
        });
    }
}
```

Chapter 84: MediaPlayer

Section 84.1: Basic creation and playing

MediaPlayer class can be used to control playback of audio/video files and streams.

Creation of MediaPlayer object can be of three types:

- Media from local resource

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

- From local URL (obtained from ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

- From external URL

```
String url = "http://....."; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Section 84.2: Media Player with Buffer progress and play position

```
public class SoundActivity extends Activity {

    private MediaPlayer mediaPlayer;
    ProgressBar progress_bar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tool_sound);
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        progress_bar = (ProgressBar) findViewById(R.id.progress_bar);

        btn_play_stop.setEnabled(false);
        btn_play_stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mediaPlayer.isPlaying()) {
                    mediaPlayer.pause();
                    btn_play_stop.setImageResource(R.drawable.ic_pause_black_24dp);
                } else {
                    mediaPlayer.start();
                }
            }
        });
    }
}
```

```

btn_play_stop.setImageResource(R.drawable.ic_play_arrow_black_24px);
    }
});

mediaPlayer.setDataSource(proxyUrl);
    mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            observer.stop();
            progress_bar.setProgress(mp.getCurrentPosition());
            // TODO Auto-generated method stub
            mediaPlayer.stop();
            mediaPlayer.reset();
        }
    });
mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
    @Override
    public void onBufferingUpdate(MediaPlayer mp, int percent) {
        progress_bar.setSecondaryProgress(percent);
    }
});
mediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
        btn_play_stop.setEnabled(true);
    }
});
observer = new MediaObserver();
    mediaPlayer.prepare();
mediaPlayer.start();
    new Thread(observer).start();
}

private MediaObserver observer = null;

private class MediaObserver implements Runnable {
    private AtomicBoolean stop = new AtomicBoolean(false);

    public void stop() {
        stop.set(true);
    }

    @Override
    public void run() {
        while (!stop.get()) {
            progress_bar.setProgress((int)((double)mediaPlayer.getCurrentPosition() /
(double)mediaPlayer.getDuration()*100));
            try {
                Thread.sleep(200);
            } catch (Exception ex) {
                Logger.log(ToolSoundActivity.this, ex);
            }
        }
    }
}

@Override
protected void onDestroy() {

```

```

        btn_play_stop.setImageResource(R.drawable.ic_play_arrow_black_24px);
    }
});

mediaPlayer.setDataSource(proxyUrl);
    mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            observer.stop();
            progress_bar.setProgress(mp.getCurrentPosition());
            // TODO Auto-generated method stub
            mediaPlayer.stop();
            mediaPlayer.reset();
        }
    });
mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
    @Override
    public void onBufferingUpdate(MediaPlayer mp, int percent) {
        progress_bar.setSecondaryProgress(percent);
    }
});
mediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
        btn_play_stop.setEnabled(true);
    }
});
observer = new MediaObserver();
    mediaPlayer.prepare();
    mediaPlayer.start();
    new Thread(observer).start();
}

private MediaObserver observer = null;

private class MediaObserver implements Runnable {
    private AtomicBoolean stop = new AtomicBoolean(false);

    public void stop() {
        stop.set(true);
    }

    @Override
    public void run() {
        while (!stop.get()) {
            progress_bar.setProgress((int)((double)mediaPlayer.getCurrentPosition() /
(double)mediaPlayer.getDuration()*100));
            try {
                Thread.sleep(200);
            } catch (Exception ex) {
                Logger.log(ToolSoundActivity.this, ex);
            }
        }
    }
}

@Override
protected void onDestroy() {

```

```

        super.onDestroy();
        mediaPlayer.stop();
    }

<LinearLayout
    android:gravity="bottom"
    android:layout_gravity="bottom"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:weightSum="1">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageButton
            app:srcCompat="@drawable/ic_play_arrow_black_24px"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:id="@+id/btn_play_stop" />

        <ProgressBar
            android:padding="8dp"
            android:progress="0"
            android:id="@+id/progress_bar"
            style="@style/Widget.AppCompat.ProgressBar.Horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />
    
```

</LinearLayout>

</LinearLayout>

第84.3节：获取系统铃声

此示例演示如何获取系统铃声的URI (RingtoneManager.TYPE_RINGTONE) :

```

private List<Uri> loadLocalRingtonesUris() {
    List<Uri> alarms = new ArrayList<>();
    try {
        RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
        ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

        Cursor alarmsCursor = ringtoneMgr.getCursor();
        int alarmsCount = alarmsCursor.getCount();
        if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
            alarmsCursor.close();
            return null;
        }

        while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
            int currentPosition = alarmsCursor.getPosition();
            alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
        }
    }
}

```

```

        super.onDestroy();
        mediaPlayer.stop();
    }

<LinearLayout
    android:gravity="bottom"
    android:layout_gravity="bottom"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:weightSum="1">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageButton
            app:srcCompat="@drawable/ic_play_arrow_black_24px"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:id="@+id/btn_play_stop" />

        <ProgressBar
            android:padding="8dp"
            android:progress="0"
            android:id="@+id/progress_bar"
            style="@style/Widget.AppCompat.ProgressBar.Horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />
    
```

</LinearLayout>

</LinearLayout>

Section 84.3: Getting system ringtones

This example demonstrates how to fetch the URI's of system ringtones (RingtoneManager.TYPE_RINGTONE):

```

private List<Uri> loadLocalRingtonesUris() {
    List<Uri> alarms = new ArrayList<>();
    try {
        RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
        ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

        Cursor alarmsCursor = ringtoneMgr.getCursor();
        int alarmsCount = alarmsCursor.getCount();
        if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
            alarmsCursor.close();
            return null;
        }

        while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
            int currentPosition = alarmsCursor.getPosition();
            alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
        }
    }
}

```

```

} catch (Exception ex) {
    ex.printStackTrace();
}

return alarms;
}

```

列表取决于请求的铃声类型。可能的类型有：

- RingtoneManager.TYPE_RINGTONE
- RingtoneManager.TYPE_NOTIFICATION
- RingtoneManager.TYPE_ALARM
- RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM

为了将铃声获取为 android.media.Ringtone，每个 Uri 必须通过 RingtoneManager 解析：

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

要播放声音，请使用以下方法：

```
public void setDataSource(Context context, Uri uri)
```

来自 android.media.MediaPlayer. MediaPlayer 必须根据 状态图 进行初始化和准备

第84.4节：异步准备

MediaPlayer\$prepare() 是一个阻塞调用，会冻结UI直到执行完成。为了解决这个问题，可以使用 MediaPlayer\$prepareAsync()。

```

mMediaPlayer = ... // 在这里初始化
mMediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener(){
    @Override
    public void onPrepared(MediaPlayer player) {
        // 当MediaPlayer准备好播放时调用
        mMediaPlayer.start();
    }
}); // 设置prepareAsync()完成时的回调
mMediaPlayer.prepareAsync(); // 异步准备，避免阻塞主线程

```

对于同步操作，错误通常通过异常或错误代码来表示，但当使用异步资源时，应确保应用程序能够适当接收到错误通知。对于MediaPlayer，

```

mMediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener(){
    @Override
    public boolean onError(MediaPlayer mp, int what, int extra) {
        // ... 适当响应 ...
        // MediaPlayer 已进入错误状态，必须重置！
        // 如果错误已被处理，则返回true
    }
});

```

第84.5节：将音频导入Android Studio并播放

这是一个示例，说明如何播放您电脑/笔记本上已有的音频文件。首先创建一个新的

```

} catch (Exception ex) {
    ex.printStackTrace();
}

return alarms;
}

```

The list depends on the types of requested ringtones. The possibilities are:

- RingtoneManager.TYPE_RINGTONE
- RingtoneManager.TYPE_NOTIFICATION
- RingtoneManager.TYPE_ALARM
- RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM

In order to get the Ringtones as android.media.Ringtone every Uri must be resolved by the RingtoneManager:

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

To play the sound, use the method:

```
public void setDataSource(Context context, Uri uri)
```

from android.media.MediaPlayer. MediaPlayer must be initialised and prepared according to the [State diagram](#)

Section 84.4: Asynchronous prepare

The MediaPlayer\$prepare() is a blocking call and will freeze the UI till execution completes. To solve this problem, MediaPlayer\$prepareAsync() can be used.

```

mMediaPlayer = ... // Initialize it here
mMediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener(){
    @Override
    public void onPrepared(MediaPlayer player) {
        // Called when the MediaPlayer is ready to play
        mMediaPlayer.start();
    }
}); // Set callback for when prepareAsync() finishes
mMediaPlayer.prepareAsync(); // Prepare asynchronously to not block the Main Thread

```

On synchronous operations, errors would normally be signaled with an exception or an error code, but whenever you use asynchronous resources, you should make sure your application is notified of errors appropriately. For MediaPlayer,

```

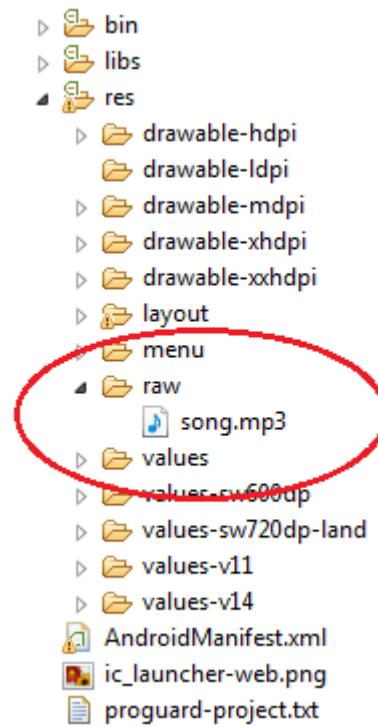
mMediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener(){
    @Override
    public boolean onError(MediaPlayer mp, int what, int extra) {
        // ... react appropriately ...
        // The MediaPlayer has moved to the Error state, must be reset!
        // Then return true if the error has been handled
    }
});

```

Section 84.5: Import audio into androidstudio and play it

This is an example how to get the play an audio file which you already have on your pc/laptop .First create a new

在 res 目录下创建一个目录，并将其命名为 raw，如下所示



将您想播放的音频复制到此文件夹中。它可以是 .mp3 或 .wav 文件。

现在例如在按钮点击时你想播放这个声音，方法如下

```
public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutapp_activity);

        MediaPlayer song=MediaPlayer.create(this, R.raw.song);

        Button button=(Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                song.start();
            }
        });
    }
}
```

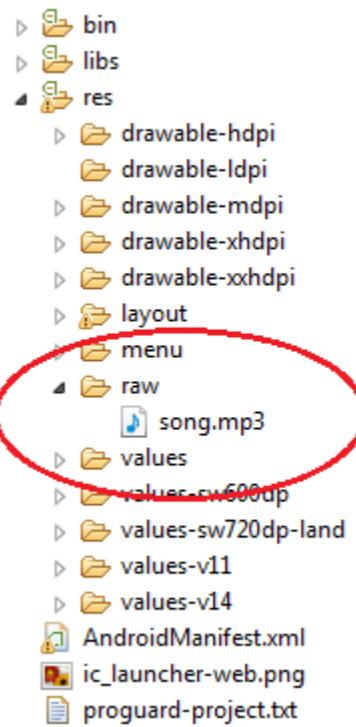
这将在按钮被点击时只播放一次歌曲，如果你想在每次点击按钮时重新播放歌曲
请写如下代码

```
public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutapp_activity);

        MediaPlayer song=MediaPlayer.create(this, R.raw.song);

        Button button=(Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
```

directory under res and name it as raw like this



copy the audio which you want to play into this folder .It may be a .mp3 or .wav file.

Now for example on button click you want to play this sound ,here is how it is done

```
public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutapp_activity);

        MediaPlayer song=MediaPlayer.create(this, R.raw.song);

        Button button=(Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                song.start();
            }
        });
    }
}
```

This will play the song only once when the button is clicked,if you want to replay the song on every button click
write code like this

```
public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutapp_activity);

        MediaPlayer song=MediaPlayer.create(this, R.raw.song);

        Button button=(Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
```

```

@Override
public void onClick(View view) {
    if (song.isPlaying()) {
        song.reset();
        song = MediaPlayer.create(getApplicationContext(), R.raw.song);
    }
    song.start();
}
}

```

第84.6节：获取和设置系统音量

音频流类型

铃声流有不同的配置文件。每个配置文件的音量都不同。

这里的每个示例都是针对AudioManager.STREAM_RING流类型编写的。但这并不是唯一的。可用的流类型有：

- STREAM_ALARM
- STREAM_DTMF
- STREAM_MUSIC
- STREAM_NOTIFICATION
- STREAM_RING
- STREAM_SYSTEM
- STREAM_VOICE_CALL

设置音量

要获取特定配置文件的音量，调用：

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);

```

当不知道流的最大值时，这个值几乎没有用处：

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);

```

这两个值的比值将给出一个相对音量 ($0 < \text{音量} < 1$)：

```
float volume = ((float) currentVolume) / streamMaxVolume
```

调节音量一步

要将流的音量提高一步，调用：

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);

```

```

@Override
public void onClick(View view) {
    if (song.isPlaying()) {
        song.reset();
        song = MediaPlayer.create(getApplicationContext(), R.raw.song);
    }
    song.start();
}
}

```

Section 84.6: Getting and setting system volume

Audio stream types

There are different profiles of ringtone streams. Each one of them has its different volume.

Every example here is written for AudioManager.STREAM_RING stream type. However this is not the only one. The available stream types are:

- STREAM_ALARM
- STREAM_DTMF
- STREAM_MUSIC
- STREAM_NOTIFICATION
- STREAM_RING
- STREAM_SYSTEM
- STREAM_VOICE_CALL

Setting volume

To get the volume of specific profile, call:

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);

```

This value is very little useful, when the maximum value for the stream is not known:

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);

```

The ratio of those two values will give a relative volume ($0 < \text{volume} < 1$):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Adjusting volume by one step

To make the volume for the stream higher by one step, call:

```

AudioManager audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);

```

要将流的音量降低一步，调用：

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);  
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

设置 MediaPlayer 使用特定的流类型

MediaPlayer类中有一个辅助函数可以实现此功能。

只需调用void setAudioStreamType(int streamtype)：

```
MediaPlayer mMedia = new MediaPlayer();  
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

To make the volume for the stream lower by one step, call:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);  
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Setting MediaPlayer to use specific stream type

There is a helper function from MediaPlayer class to do this.

Just call **void** setAudioStreamType(**int** streamtype):

```
MediaPlayer mMedia = new MediaPlayer();  
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

第85章：Android声音与媒体

第85.1节：如何为API >19选择图片和视频

这里有一段经过测试的图片和视频代码。它适用于所有小于19和大于19的API版本。

图片：

```
if (Build.VERSION.SDK_INT <= 19) {
    Intent i = new Intent();
    i.setType("image/*");
    i.setAction(Intent.ACTION_GET_CONTENT);
    i.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(i, 10);
} else if (Build.VERSION.SDK_INT > 19) {
    Intent intent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 10);
}
```

视频：

```
if (Build.VERSION.SDK_INT <= 19) {
    Intent i = new Intent();
    i.setType("video/*");
    i.setAction(Intent.ACTION_GET_CONTENT);
    i.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(i, 20);
} else if (Build.VERSION.SDK_INT > 19) {
    Intent intent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 20);
}
```

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK) {

        if (requestCode == 10) {
            Uri selectedImageUri = data.getData();
            String selectedImagePath = getRealPathFromURI(selectedImageUri);
        } else if (requestCode == 20) {
            Uri selectedVideoUri = data.getData();
            String selectedVideoPath = getRealPathFromURI(selectedVideoUri);
        }

        public String getRealPathFromURI(Uri uri) {
            if (uri == null) {
                return null;
            }
            String[] projection = {MediaStore.Images.Media.DATA};
            Cursor cursor = getActivity().getContentResolver().query(uri, projection, null, null, null);
            if (cursor != null) {
                int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
```

Chapter 85: Android Sound and Media

Section 85.1: How to pick image and video for api >19

Here is a tested code for image and video. It will work for all APIs less than 19 and greater than 19 as well.

Image:

```
if (Build.VERSION.SDK_INT <= 19) {
    Intent i = new Intent();
    i.setType("image/*");
    i.setAction(Intent.ACTION_GET_CONTENT);
    i.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(i, 10);
} else if (Build.VERSION.SDK_INT > 19) {
    Intent intent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 10);
}
```

Video:

```
if (Build.VERSION.SDK_INT <= 19) {
    Intent i = new Intent();
    i.setType("video/*");
    i.setAction(Intent.ACTION_GET_CONTENT);
    i.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(i, 20);
} else if (Build.VERSION.SDK_INT > 19) {
    Intent intent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 20);
}
```

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK) {

        if (requestCode == 10) {
            Uri selectedImageUri = data.getData();
            String selectedImagePath = getRealPathFromURI(selectedImageUri);
        } else if (requestCode == 20) {
            Uri selectedVideoUri = data.getData();
            String selectedVideoPath = getRealPathFromURI(selectedVideoUri);
        }

        public String getRealPathFromURI(Uri uri) {
            if (uri == null) {
                return null;
            }
            String[] projection = {MediaStore.Images.Media.DATA};
            Cursor cursor = getActivity().getContentResolver().query(uri, projection, null, null, null);
            if (cursor != null) {
                int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
```

```

cursor.moveToFirst();
    return cursor.getString(column_index);
}
return uri.getPath();
}

```

第85.2节：通过SoundPool播放声音

```

public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;

    /** 活动首次创建时调用. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View view = findViewById(R.id.textView1);
        view.setOnTouchListener(this);
        // 设置硬件按钮控制音乐
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
        // 加载声音
        soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
        soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
            @Override
            public void onLoadComplete(SoundPool soundPool, int sampleId,
                                       int status) {
                loaded = true;
            }
        });
        soundID = soundPool.load(this, R.raw.sound1, 1);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            // 获取用户的声音设置
            AudioManager audioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
            float actualVolume = (float) audioManager
                .getStreamVolume(AudioManager.STREAM_MUSIC);
            float maxVolume = (float) audioManager
                .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
            float volume = actualVolume / maxVolume;
            // 声音是否已经加载?
            if (loaded) {
                soundPool.play(soundID, volume, volume, 1, 0, 1f);
                Log.e("Test", "Played sound");
            }
        }
        return false;
    }
}

```

```

cursor.moveToFirst();
    return cursor.getString(column_index);
}
return uri.getPath();
}

```

Section 85.2: Play sounds via SoundPool

```

public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View view = findViewById(R.id.textView1);
        view.setOnTouchListener(this);
        // Set the hardware buttons to control the music
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
        // Load the sound
        soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
        soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
            @Override
            public void onLoadComplete(SoundPool soundPool, int sampleId,
                                       int status) {
                loaded = true;
            }
        });
        soundID = soundPool.load(this, R.raw.sound1, 1);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            // Getting the user sound settings
            AudioManager audioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
            float actualVolume = (float) audioManager
                .getStreamVolume(AudioManager.STREAM_MUSIC);
            float maxVolume = (float) audioManager
                .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
            float volume = actualVolume / maxVolume;
            // Is the sound loaded already?
            if (loaded) {
                soundPool.play(soundID, volume, volume, 1, 0, 1f);
                Log.e("Test", "Played sound");
            }
        }
        return false;
    }
}

```

第86章：MediaSession

第86.1节：接收和处理按钮事件

此示例在启动Service时创建一个MediaSession对象。当Service被销毁时，MediaSession对象也会被释放：

```
public final class MyService extends Service {
    private static MediaSession s_mediaSession;

    @Override
    public void onCreate() {
        // 实例化新的 MediaSession 对象。
        configureMediaSession();
    }

    @Override
    public void onDestroy() {
        if (s_mediaSession != null)
            s_mediaSession.release();
    }
}
```

以下方法实例化并配置了MediaSession按钮回调：

```
private void 配置媒体会话 {
    s_mediaSession = new MediaSession(this, "MyMediaSession");

    // MediaSession.Callback 类中重写的方法。
    s_mediaSession.setCallback(new MediaSession.Callback() {
        @Override
        public boolean onMediaButtonEvent(Intent mediaButtonIntent) {
            Log.d(TAG, "onMediaButtonEvent 调用: " + mediaButtonIntent);
            KeyEvent ke = mediaButtonIntent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (ke != null && ke.getAction() == KeyEvent.ACTION_DOWN) {
                int keyCode = ke.getKeyCode();
                Log.d(TAG, "onMediaButtonEvent 收到命令: " + ke);
            }
            return super.onMediaButtonEvent(mediaButtonIntent);
        }

        @Override
        public void onSkipToNext() {
            Log.d(TAG, "onSkipToNext 调用 (媒体按钮被按下)");
            Toast.makeText(getApplicationContext(), "onSkipToNext 调用",
                    Toast.LENGTH_SHORT).show();
            skipToNextPlaylistItem(); // 处理此按钮按下事件。
            super.onSkipToNext();
        }

        @Override
        public void onSkipToPrevious() {
            Log.d(TAG, "onSkipToPrevious 调用 (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToPrevious 调用",
                    Toast.LENGTH_SHORT).show();
            skipToPreviousPlaylistItem(); // 处理此按钮按下事件。
            super.onSkipToPrevious();
        }
    });
}
```

Chapter 86: MediaSession

Section 86.1: Receiving and handling button events

This example creates a [MediaSession](#) object when a [Service](#) is started. The MediaSession object is released when the Service gets destroyed:

```
public final class MyService extends Service {
    private static MediaSession s_mediaSession;

    @Override
    public void onCreate() {
        // Instantiate new MediaSession object.
        configureMediaSession();
    }

    @Override
    public void onDestroy() {
        if (s_mediaSession != null)
            s_mediaSession.release();
    }
}
```

The following method instantiates and configures the MediaSession button callbacks:

```
private void configureMediaSession {
    s_mediaSession = new MediaSession(this, "MyMediaSession");

    // Overridden methods in the MediaSession.Callback class.
    s_mediaSession.setCallback(new MediaSession.Callback() {
        @Override
        public boolean onMediaButtonEvent(Intent mediaButtonIntent) {
            Log.d(TAG, "onMediaButtonEvent called: " + mediaButtonIntent);
            KeyEvent ke = mediaButtonIntent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (ke != null && ke.getAction() == KeyEvent.ACTION_DOWN) {
                int keyCode = ke.getKeyCode();
                Log.d(TAG, "onMediaButtonEvent Received command: " + ke);
            }
            return super.onMediaButtonEvent(mediaButtonIntent);
        }

        @Override
        public void onSkipToNext() {
            Log.d(TAG, "onSkipToNext called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToNext called",
                    Toast.LENGTH_SHORT).show();
            skipToNextPlaylistItem(); // Handle this button press.
            super.onSkipToNext();
        }

        @Override
        public void onSkipToPrevious() {
            Log.d(TAG, "onSkipToPrevious called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToPrevious called",
                    Toast.LENGTH_SHORT).show();
            skipToPreviousPlaylistItem(); // Handle this button press.
            super.onSkipToPrevious();
        }
    });
}
```

```

@Override
public void onPause() {
Log.d(TAG, "onPause called (media button pressed)");
Toast.makeText(getApplicationContext(), "onPause called", Toast.LENGTH_SHORT).show();
mpPause(); // 暂停播放器。
super.onPause();
}

@Override
public void onPlay() {
Log.d(TAG, "onPlay called (media button pressed)");
mpStart(); // 启动播放器/播放。
super.onPlay();
}

@Override
public void onStop() {
Log.d(TAG, "onStop 被调用 (媒体按钮被按下) ");
mpReset(); // 停止和/或重置播放器。
super.onStop();
}

s_mediaSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS |
MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
s_mediaSession.setActive(true);
}

```

以下方法使用 A2DP 向设备发送元数据（存储在 `HashMap` 中）：

```

void sendMetaData(@NonNull final HashMap<String, String> hm) {
// 如果未使用蓝牙 A2DP，则返回。
if (!((AudioManager) getSystemService(Context.AUDIO_SERVICE)).isBluetoothA2dpOn()) return;

MediaMetadata metadata = new MediaMetadata.Builder()
.putString(MediaMetadata.METADATA_KEY_TITLE, hm.get("Title"))
.putString(MediaMetadata.METADATA_KEY_ALBUM, hm.get("Album"))
.putString(MediaMetadata.METADATA_KEY_ARTIST, hm.get("Artist"))
.putString(MediaMetadata.METADATA_KEY_AUTHOR, hm.get("Author"))
.putString(MediaMetadata.METADATA_KEY_COMPOSER, hm.get("Composer"))
.putString(MediaMetadata.METADATA_KEY_WRITER, hm.get("Writer"))
.putString(MediaMetadata.METADATA_KEY_DATE, hm.get("Date"))
.putString(MediaMetadata.METADATA_KEY_GENRE, hm.get("Genre"))
.putLong(MediaMetadata.METADATA_KEY_YEAR, tryParse(hm.get("Year")))
.putLong(MediaMetadata.METADATA_KEY_DURATION, tryParse(hm.get("Raw Duration")))
.putLong(MediaMetadata.METADATA_KEY_TRACK_NUMBER, tryParse(hm.get("Track Number")))
.build();

s_mediaSession.setMetadata(metadata);
}

```

以下方法设置了 `PlaybackState`。它还设置了 `MediaSession` 将响应的按钮操作：

```

private void setPlaybackState(@NonNull final int stateValue) {
PlaybackState state = new PlaybackState.Builder()
.setActions(PlaybackState.ACTION_PLAY | PlaybackState.ACTION_SKIP_TO_NEXT
| PlaybackState.ACTION_PAUSE | PlaybackState.ACTION_SKIP_TO_PREVIOUS
| PlaybackState.ACTION_STOP | PlaybackState.ACTION_PLAY_PAUSE)
.setState(stateValue, PlaybackState.PLAYBACK_POSITION_UNKNOWN, 0)
.build();
}

```

```

@Override
public void onPause() {
Log.d(TAG, "onPause called (media button pressed)");
Toast.makeText(getApplicationContext(), "onPause called", Toast.LENGTH_SHORT).show();
mpPause(); // Pause the player.
super.onPause();
}

@Override
public void onPlay() {
Log.d(TAG, "onPlay called (media button pressed)");
mpStart(); // Start player/playback.
super.onPlay();
}

@Override
public void onStop() {
Log.d(TAG, "onStop called (media button pressed)");
mpReset(); // Stop and/or reset the player.
super.onStop();
}

s_mediaSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS |
MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
s_mediaSession.setActive(true);
}

```

The following method sends meta data (stored in a `HashMap`) to the device using A2DP:

```

void sendMetaData(@NonNull final HashMap<String, String> hm) {
// Return if Bluetooth A2DP is not in use.
if (!((AudioManager) getSystemService(Context.AUDIO_SERVICE)).isBluetoothA2dpOn()) return;

MediaMetadata metadata = new MediaMetadata.Builder()
.putString(MediaMetadata.METADATA_KEY_TITLE, hm.get("Title"))
.putString(MediaMetadata.METADATA_KEY_ALBUM, hm.get("Album"))
.putString(MediaMetadata.METADATA_KEY_ARTIST, hm.get("Artist"))
.putString(MediaMetadata.METADATA_KEY_AUTHOR, hm.get("Author"))
.putString(MediaMetadata.METADATA_KEY_COMPOSER, hm.get("Composer"))
.putString(MediaMetadata.METADATA_KEY_WRITER, hm.get("Writer"))
.putString(MediaMetadata.METADATA_KEY_DATE, hm.get("Date"))
.putString(MediaMetadata.METADATA_KEY_GENRE, hm.get("Genre"))
.putLong(MediaMetadata.METADATA_KEY_YEAR, tryParse(hm.get("Year")))
.putLong(MediaMetadata.METADATA_KEY_DURATION, tryParse(hm.get("Raw Duration")))
.putLong(MediaMetadata.METADATA_KEY_TRACK_NUMBER, tryParse(hm.get("Track Number")))
.build();

s_mediaSession.setMetadata(metadata);
}

```

The following method sets the `PlaybackState`. It also sets which button actions the `MediaSession` will respond to:

```

private void setPlaybackState(@NonNull final int stateValue) {
PlaybackState state = new PlaybackState.Builder()
.setActions(PlaybackState.ACTION_PLAY | PlaybackState.ACTION_SKIP_TO_NEXT
| PlaybackState.ACTION_PAUSE | PlaybackState.ACTION_SKIP_TO_PREVIOUS
| PlaybackState.ACTION_STOP | PlaybackState.ACTION_PLAY_PAUSE)
.setState(stateValue, PlaybackState.PLAYBACK_POSITION_UNKNOWN, 0)
.build();
}

```

```
s_mediaSession.setPlaybackState(state);  
}
```

```
s_mediaSession.setPlaybackState(state);  
}
```

第87章：MediaStore

第87.1节：从设备的特定文件夹获取音频/MP3文件或获取所有文件

首先，在项目的清单文件中添加以下权限，以启用设备存储访问：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

然后，创建文件*AudioModel.class*，并将以下模型类放入其中，以便获取和设置列表项：

```
public class AudioModel {
    String aPath;
    String aName;
    String aAlbum;
    String aArtist;

    public String getaPath() {
        return aPath;
    }
    public void setaPath(String aPath) {
        this.aPath = aPath;
    }
    public String getaName() {
        return aName;
    }
    public void setaName(String aName) {
        this.aName = aName;
    }
    public String getaAlbum() {
        return aAlbum;
    }
    public void setaAlbum(String aAlbum) {
        this.aAlbum = aAlbum;
    }
    public String getaArtist() {
        return aArtist;
    }
    public void setaArtist(String aArtist) {
        this.aArtist = aArtist;
    }
}
```

接下来，使用以下方法从设备的文件夹中读取所有MP3文件，或读取设备中的所有文件：

```
public List<AudioModel> getAllAudioFromDevice(final Context context) {
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection = {MediaStore.Audio.AudioColumns.DATA, MediaStore.Audio.AudioColumns.TITLE,
    MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA + " like ? ", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
```

Chapter 87: MediaStore

Section 87.1: Fetch Audio/MP3 files from specific folder of device or fetch all files

First, add the following permissions to the manifest of your project in order to enable device storage access:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Then, create the file *AudioModel.class* and put the following model class into it in order to allow getting and setting list items:

```
public class AudioModel {
    String aPath;
    String aName;
    String aAlbum;
    String aArtist;

    public String getaPath() {
        return aPath;
    }
    public void setaPath(String aPath) {
        this.aPath = aPath;
    }
    public String getaName() {
        return aName;
    }
    public void setaName(String aName) {
        this.aName = aName;
    }
    public String getaAlbum() {
        return aAlbum;
    }
    public void setaAlbum(String aAlbum) {
        this.aAlbum = aAlbum;
    }
    public String getaArtist() {
        return aArtist;
    }
    public void setaArtist(String aArtist) {
        this.aArtist = aArtist;
    }
}
```

Next, use the following method to read all MP3 files from a folder of your device or to read all files of your device:

```
public List<AudioModel> getAllAudioFromDevice(final Context context) {
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection = {MediaStore.Audio.AudioColumns.DATA, MediaStore.Audio.AudioColumns.TITLE,
    MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA + " like ? ", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
```

```

AudioModel audioModel = new AudioModel();
    String path = c.getString(0);
    String name = c.getString(1);
    String album = c.getString(2);
    String artist = c.getString(3);

audioModel.setaName(name);
    audioModel.setaAlbum(album);
    audioModel.setaArtist(artist);
    audioModel.setaPath(path);

    Log.e("Name :" + name, " Album :" + album);
    Log.e("Path :" + path, " Artist :" + artist);

    tempAudioList.add(audioModel);
}

c.close();
}

return tempAudioList;
}

```

以上代码将返回包含音乐名称、路径、艺术家和专辑的所有MP3文件列表。更多详情请参阅[Media.Store.Audio](#)文档。

为了读取特定文件夹的文件，请使用以下查询（需要替换文件夹名称）：

```

Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ?",
    new String[]{"%yourFolderName%"}, // 在此处填写您的设备文件夹/文件位置.
    null);

```

如果您想检索设备上的所有文件，请使用以下查询：

```

Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);

```

注意：别忘了启用存储访问权限。

现在，您只需调用上述方法即可获取MP3文件：

```
getAllAudioFromDevice(this);
```

带Activity的示例

```

public class ReadAudioFilesActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audio_list);

        /**
         * 这将返回所有MP3文件的列表。使用该列表来显示数据。
         */
        getAllAudioFromDevice(this);
    }
}

```

```

AudioModel audioModel = new AudioModel();
    String path = c.getString(0);
    String name = c.getString(1);
    String album = c.getString(2);
    String artist = c.getString(3);

audioModel.setaName(name);
    audioModel.setaAlbum(album);
    audioModel.setaArtist(artist);
    audioModel.setaPath(path);

    Log.e("Name :" + name, " Album :" + album);
    Log.e("Path :" + path, " Artist :" + artist);

    tempAudioList.add(audioModel);
}

c.close();
}

return tempAudioList;
}

```

The code above will return a list of all MP3 files with the music's name, path, artist, and album. For more details please refer to the [Media.Store.Audio](#) documentation.

In order to read files of a specific folder, use the following query (you need to replace the folder name):

```

Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ?",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);

```

If you want to retrieve all files from your device, then use the following query:

```

Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);

```

Note: Don't forget to enable storage access permissions.

Now, all you have to do is to call the method above in order to get the MP3 files:

```
getAllAudioFromDevice(this);
```

Example with Activity

```

public class ReadAudioFilesActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audio_list);

        /**
         * This will return a list of all MP3 files. Use the list to display data.
         */
        getAllAudioFromDevice(this);
    }
}

```

```

// 读取所有音频/MP3文件的方法。
public List<AudioModel> getAllAudioFromDevice(final Context context) {
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection =
    {MediaStore.Audio.AudioColumns.DATA, MediaStore.Audio.AudioColumns.TITLE
    , MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA
    + " like ?", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
            // 创建一个模型对象。
            AudioModel audioModel = new AudioModel();

            String path = c.getString(0); // 获取路径。
            String name = c.getString(1); // 获取名称。
            String album = c.getString(2); // 获取专辑名称。
            String artist = c.getString(3); // 获取艺术家名称。

            // 将数据设置到模型对象。
            audioModel.setaName(name);
            audioModel.setaAlbum(album);
            audioModel.setaArtist(artist);
            audioModel.setaPath(path);

            Log.e("Name :" + name, " Album :" + album);
            Log.e("Path :" + path, " Artist :" + artist);

            // 将模型对象添加到列表中。
            tempAudioList.add(audioModel);
        }
        c.close();
    }

    // 返回列表。
    return tempAudioList;
}

```

```

// Method to read all the audio/MP3 files.
public List<AudioModel> getAllAudioFromDevice(final Context context) {
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection =
    {MediaStore.Audio.AudioColumns.DATA, MediaStore.Audio.AudioColumns.TITLE
    , MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA
    + " like ?", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
            // Create a model object.
            AudioModel audioModel = new AudioModel();

            String path = c.getString(0); // Retrieve path.
            String name = c.getString(1); // Retrieve name.
            String album = c.getString(2); // Retrieve album name.
            String artist = c.getString(3); // Retrieve artist name.

            // Set data to the model object.
            audioModel.setaName(name);
            audioModel.setaAlbum(album);
            audioModel.setaArtist(artist);
            audioModel.setaPath(path);

            Log.e("Name :" + name, " Album :" + album);
            Log.e("Path :" + path, " Artist :" + artist);

            // Add the model object to the list .
            tempAudioList.add(audioModel);
        }
        c.close();
    }

    // Return the list.
    return tempAudioList;
}

```

第88章：Multidex与Dex方法限制

DEX指的是Android应用程序（APK）的可执行字节码文件，形式为Dalvik可执行文件（DEX），其中包含用于运行应用程序的编译代码。

Dalvik可执行文件规范限制单个DEX文件中可引用的方法总数为65,536（64K）——包括Android框架方法、库方法以及您自己的代码中的方法。

要突破此限制，需要配置应用程序的构建过程以生成多个DEX文件，这称为Multidex。

第88.1节：启用Multidex

为了启用Multidex配置，您需要：

- 更改您的Gradle构建配置
- 使用MultiDexApplication或在您的Application类中启用MultiDex

Gradle配置

在app/build.gradle中添加以下内容：

```
android {  
    compileSdkVersion 24  
    buildToolsVersion "24.0.1"  
  
    defaultConfig {  
        ...  
        minSdkVersion 14  
        targetSdkVersion 24  
        ...  
  
        // 启用 multidex 支持。  
        multiDexEnabled true  
    }  
    ...  
  
}  
  
dependencies {  
    compile 'com.android.support:multidex:1.0.1'  
}
```

在您的应用中启用 MultiDex

然后选择以下三种方式之一：

- 通过继承 Application 实现 Multidex
- 通过继承 MultiDexApplication 实现 Multidex
- 直接使用 MultiDexApplication 实现 Multidex

当这些配置设置添加到应用中时，Android 构建工具会构建一个主 dex (classes.dex) 和根据需要的辅助 dex (classes2.dex, classes3.dex)。构建系统随后会将它们打包成一个 APK 文件以供分发。

Chapter 88: Multidex and the Dex Method Limit

DEX means Android app's (APK) executable bytecode files in the form of Dalvik Executable (DEX) files, which contain the compiled code used to run your app.

The Dalvik Executable specification limits the total number of methods that can be referenced within a single DEX file to 65,536 (64K)—including Android framework methods, library methods, and methods in your own code.

To overcome this limit requires configure your app build process to generate more than one DEX file, known as a Multidex.

Section 88.1: Enabling Multidex

In order to enable a multidex configuration you need:

- to change your Gradle build configuration
- to use a MultiDexApplication or enable the MultiDex in your Application class

Gradle configuration

In app/build.gradle add these parts:

```
android {  
    compileSdkVersion 24  
    buildToolsVersion "24.0.1"  
  
    defaultConfig {  
        ...  
        minSdkVersion 14  
        targetSdkVersion 24  
        ...  
  
        // Enabling multidex support.  
        multiDexEnabled true  
    }  
    ...  
  
}  
  
dependencies {  
    compile 'com.android.support:multidex:1.0.1'  
}
```

Enable MultiDex in your Application

Then proceed with one of three options:

- Multidex by extending Application
- Multidex by extending MultiDexApplication
- Multidex by using MultiDexApplication directly

When these configuration settings are added to an app, the Android build tools construct a primary dex (classes.dex) and supporting (classes2.dex, classes3.dex) as needed. The build system will then package them into an APK file for distribution.

第88.2节：通过扩展Application实现Multidex

如果您的项目需要一个Application子类，请使用此选项。

在清单文件的application标签中，使用android:name属性指定此Application子类。

在Application子类中，添加attachBaseContext()方法的重写，并在该方法中调用MultiDex.install()：

```
包com.example;

导入 android.app.Application;
导入 android.content.Context;

/**
 * 支持multidex的扩展应用程序
 */
公共类MyApplication extends Application {

    @Override
    受保护的void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

确保在您的AndroidManifest.xml的application标签中指定了Application子类：

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

第88.3节：通过继承MultiDexApplication实现Multidex

这与使用Application子类并重写attachBaseContext()方法非常相似。

但是，使用此方法时，无需重写attachBaseContext()，因为这已经在MultiDexApplication超类中完成。

继承MultiDexApplication而不是Application：

```
包名com.example;

导入 android.support.multidex.MultiDexApplication;
导入 android.content.Context;

/**
 * 继承自MultiDexApplication
 */
公共类MyApplication extends MultiDexApplication {

    // 无需重写attachBaseContext()
    //.....
}
```

Section 88.2: Multidex by extending Application

Use this option if your project requires an Application subclass.

Specify this Application subclass using the android:name property in the manifest file inside the application tag.

In the Application subclass, add the attachBaseContext() method override, and in that method call MultiDex.install():

```
package com.example;

import android.app.Application;
import android.content.Context;

/**
 * Extended application that support multidex
 */
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

Ensure that the Application subclass is specified in the application tag of your AndroidManifest.xml:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Section 88.3: Multidex by extending MultiDexApplication

This is very similar to using an Application subclass and overriding the attachBaseContext() method.

However, using this method, you don't need to override attachBaseContext() as this is already done in the MultiDexApplication superclass.

Extend MultiDexApplication instead of Application:

```
package com.example;

import android.support.multidex.MultiDexApplication;
import android.content.Context;

/**
 * Extended MultiDexApplication
 */
public class MyApplication extends MultiDexApplication {

    // No need to override attachBaseContext()

    //.....
}
```

将此类添加到您的 AndroidManifest.xml 中，方法与扩展 Application 类完全相同：

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
```

第88.4节：通过直接使用 MultiDexApplication 实现多重dex

如果您不需要自定义的Application子类，请使用此选项。

这是最简单的选项，但这样您无法提供自己的Application子类。如果需要Application子类，则必须切换到其他选项之一来实现。

对于此选项，只需在 AndroidManifest.xml 中的application标签的android:name属性中指定完全限定类名android.support.multidex.MultiDexApplication：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.multidex.myapplication">
    <application
        ...
        android:name="android.support.multidex.MultiDexApplication"
        ...
    </application>
</manifest>
```

第88.5节：每次构建时统计方法引用数 (Dexcount Gradle 插件)

dexcount 插件在构建成功后统计方法和类资源数量。

在app/build.gradle中添加插件：

```
apply plugin: 'com.android.application'

buildscript {
    repositories {
        mavenCentral() // 或者 jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

在app/build.gradle文件中应用插件：

```
apply plugin: 'com.getkeepsafe.dexcount'
```

查找插件生成的输出数据位置：

..../app/build/outputs/dexcount

特别有用的是位于以下位置的.html图表：

Add this class to your AndroidManifest.xml exactly as if you were extending Application:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Section 88.4: Multidex by using MultiDexApplication directly

Use this option if you don't need an Application subclass.

This is the simplest option, but this way you can't provide your own Application subclass. If an Application subclass is needed, you will have to switch to one of the other options to do so.

For this option, simply specify the fully-qualified class name android.support.multidex.MultiDexApplication for the android:name property of the application tag in the AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.multidex.myapplication">
    <application
        ...
        android:name="android.support.multidex.MultiDexApplication"
        ...
    </application>
</manifest>
```

Section 88.5: Counting Method References On Every Build (Dexcount Gradle Plugin)

The [dexcount plugin](#) counts methods and class resource count after a successful build.

Add the plugin in the app/build.gradle:

```
apply plugin: 'com.android.application'

buildscript {
    repositories {
        mavenCentral() // 或者 jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Apply the plugin in the app/build.gradle file:

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Look for the output data generated by the plugin in:

..../app/build/outputs/dexcount

Especially useful is the .html chart in:

第89章：使用同步适配器进行数据同步

第89.1节：带有存根提供者的虚拟同步适配器

同步适配器

```
/**  
 * 为应用定义一个同步适配器。  
 * <p/>  
 * <p>该类在{@link SyncService}中实例化，SyncService也将同步适配器绑定到系统。  
 * SyncAdapter 应该只在 SyncService 中初始化，绝不应在其他任何地方初始化。  
 * <p/>  
 * 系统通过 SyncService 提供的 IBinder 对象，通过 RPC 调用调用 onPerformSync() 方法。  
  
 */  
类 SyncAdapter 继承 AbstractThreadedSyncAdapter {  
    /**  
     * 构造函数。获取内容解析器的句柄以供后续使用。  
     */  
    public SyncAdapter(Context context, boolean autoInitialize) {  
        super(context, autoInitialize);  
    }  
  
    /**  
     * 构造函数。获取内容解析器的句柄以供后续使用。  
     */  
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {  
        super(context, autoInitialize, allowParallelSyncs);  
    }  
  
    @Override  
    public void onPerformSync(Account account, Bundle extras, String authority,  
                             ContentProviderClient provider, SyncResult syncResult) {  
        // 你想在后台执行的任务。  
        Log.e("" + account.name, "同步开始");  
    }  
}
```

同步服务

```
/**  
 * 定义一个服务，返回同步适配器类的 IBinder，允许同步适配器  
 * 框架调用 * onPerformSync() 方法。  
  
 */  
public class SyncService extends Service {  
    // 同步适配器实例的存储  
    private static SyncAdapter sSyncAdapter = null;  
    // 用作线程安全锁的对象  
    private static final Object sSyncAdapterLock = new Object();  
  
    /*  
     * 实例化同步适配器对象。  
     */  
    @Override  
    public void onCreate() {  
        /*  
         * 将同步适配器创建为单例。  
         * 设置同步适配器为可同步
    }
```

Chapter 89: Data Synchronization with Sync Adapter

Section 89.1: Dummy Sync Adapter with Stub Provider

SyncAdapter

```
/**  
 * Define a sync adapter for the app.  
 * <p>  
 * <p>This class is instantiated in {@link SyncService}，which also binds SyncAdapter to the system.  
 * SyncAdapter should only be initialized in SyncService, never anywhere else.  
 * <p>  
 * <p>The system calls onPerformSync() via an RPC call through the IBinder object supplied by  
 * SyncService.  
 */  
class SyncAdapter extends AbstractThreadedSyncAdapter {  
    /**  
     * Constructor. Obtains handle to content resolver for later use.  
     */  
    public SyncAdapter(Context context, boolean autoInitialize) {  
        super(context, autoInitialize);  
    }  
  
    /**  
     * Constructor. Obtains handle to content resolver for later use.  
     */  
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {  
        super(context, autoInitialize, allowParallelSyncs);  
    }  
  
    @Override  
    public void onPerformSync(Account account, Bundle extras, String authority,  
                            ContentProviderClient provider, SyncResult syncResult) {  
        //Jobs you want to perform in background.  
        Log.e("" + account.name, "Sync Start");  
    }  
}
```

Sync Service

```
/**  
 * Define a Service that returns an IBinder for the  
 * sync adapter class, allowing the sync adapter framework to call  
 * onPerformSync().  
 */  
public class SyncService extends Service {  
    // Storage for an instance of the sync adapter  
    private static SyncAdapter sSyncAdapter = null;  
    // Object to use as a thread-safe lock  
    private static final Object sSyncAdapterLock = new Object();  
  
    /*  
     * Instantiate the sync adapter object.  
     */  
    @Override  
    public void onCreate() {  
        /*  
         * Create the sync adapter as a singleton.  
         * Set the sync adapter as syncable
    }
```

```

* 不允许并行同步
 */
synchronized (sSyncAdapterLock) {
    if (sSyncAdapter == null) {
        sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
    }
}

/**
* 返回一个允许系统调用同步适配器的对象。
*/
@Override
public IBinder onBind(Intent intent) {
    /*
    * 获取允许外部进程调用 onPerformSync() 的对象。该对象在同步适配器构造函数调用 super() 时由基类代码创建。
    */

    /*
    return sSyncAdapter.getSyncAdapterBinder();
}

```

认证器

```

public class 认证器 extends AbstractAccountAuthenticator {
    // 简单构造函数
    public 认证器(Context context) {
        super(context);
    }

    // 不支持编辑属性
    @Override
    public Bundle editProperties(
        AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }

    // 不要添加额外的账户
    @Override
    public Bundle addAccount(
        AccountAuthenticatorResponse r,
        String s,
        String s2,
        String[] strings,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // 忽略确认凭证的尝试
    @Override
    public Bundle confirmCredentials(
        AccountAuthenticatorResponse r,
        Account account,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // 不支持获取认证令牌
    @Override
    public Bundle getAuthToken(

```

```

* Disallow parallel syncs
*/
synchronized (sSyncAdapterLock) {
    if (sSyncAdapter == null) {
        sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
    }
}

/**
* Return an object that allows the system to invoke
* the sync adapter.
*/
@Override
public IBinder onBind(Intent intent) {
    /*
    * Get the object that allows external processes
    * to call onPerformSync(). The object is created
    * in the base class code when the SyncAdapter
    * constructors call super()
    */
    return sSyncAdapter.getSyncAdapterBinder();
}

Authenticator
public class Authenticator extends AbstractAccountAuthenticator {
    // Simple constructor
    public Authenticator(Context context) {
        super(context);
    }

    // Editing properties is not supported
    @Override
    public Bundle editProperties(
        AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }

    // Don't add additional accounts
    @Override
    public Bundle addAccount(
        AccountAuthenticatorResponse r,
        String s,
        String s2,
        String[] strings,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
        AccountAuthenticatorResponse r,
        Account account,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Getting an authentication token is not supported
    @Override
    public Bundle getAuthToken(

```

```

AccountAuthenticatorResponse r,
    Account account,
    String s,
Bundle bundle) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}

// 获取认证令牌标签不被支持
@Override
public String getAuthTokenLabel(String s) {
    throw new UnsupportedOperationException();
}

// 更新用户凭据不被支持
@Override
public Bundle updateCredentials(
    AccountAuthenticatorResponse r,
    Account account,
    String s, Bundle bundle) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}

// 检查账户功能不被支持
@Override
public Bundle hasFeatures(
AccountAuthenticatorResponse r,
    Account account, String[] strings) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}
}

```

认证服务

```

/***
 * 一个绑定服务，在启动时实例化认证器
 */

public class AuthenticatorService extends Service {
    // 存储认证器对象的实例字段
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // 创建一个新的认证器对象
        mAuthenticator = new Authenticator(this);
    }
    /*
     * 当系统绑定到此服务以进行RPC调用时
     * 返回认证器的IBinder。
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}

```

AndroidManifest.xml 添加内容

```

<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

<service
    android:name=".syncAdapter.SyncService"

```

```

AccountAuthenticatorResponse r,
    Account account,
    String s,
Bundle bundle) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}

// Getting a label for the auth token is not supported
@Override
public String getAuthTokenLabel(String s) {
    throw new UnsupportedOperationException();
}

// Updating user credentials is not supported
@Override
public Bundle updateCredentials(
    AccountAuthenticatorResponse r,
    Account account,
    String s, Bundle bundle) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}

// Checking features for the account is not supported
@Override
public Bundle hasFeatures(
    AccountAuthenticatorResponse r,
    Account account, String[] strings) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}
}

```

Authenticator Service

```

/***
 * A bound Service that instantiates the authenticator
 * when started.
 */
public class AuthenticatorService extends Service {
    // Instance field that stores the authenticator object
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new Authenticator(this);
    }
    /*
     * When the system binds to this Service to make the RPC call
     * return the authenticator's IBinder.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}

```

AndroidManifest.xml additions

```

<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

<service
    android:name=".syncAdapter.SyncService"

```

```

        android:exported="true"
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />

```

res/xml/authenticator.xml

```

<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />

```

res/xml/syncadapter.xml

```

<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
    android:userVisible="false" />

```

StubProvider

```

/*
 * 定义一个ContentProvider的实现，所有方法均为存根
 *
 */
public class StubProvider extends ContentProvider {
    /*
     * 始终返回 true，表示提供者已正确加载。
     *
     * @Override
     public boolean onCreate() {
         return true;
     }

     /*
     * 对于 MIME 类型返回无类型
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }
}

```

```

        android:exported="true"
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />

```

res/xml/authenticator.xml

```

<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />

```

res/xml/syncadapter.xml

```

<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
    android:userVisible="false" />

```

StubProvider

```

/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    /*
     * Return no type for MIME type
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }
}

```

```

}

/*
* query() 始终返回无结果
*
*/
@Override
public Cursor query(
    Uri uri,
    String[] projection,
    String selection,
    String[] selectionArgs,
    String sortOrder) {
    return null;
}

/*
* insert() 始终返回 null (无 URI)
*/
@Override
public Uri insert(Uri uri, ContentValues values) {
    return null;
}

/*
* delete() 总是返回“无行受影响” (0)
*/
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    return 0;
}

/*
* update() 总是返回“无行受影响” (0)
*/
public int update(
Uri uri,
ContentValues values,
String selection,
String[] selectionArgs) {
    return 0;
}
}

```

登录成功后调用此函数以使用登录的用户ID创建账户

```

public Account CreateSyncAccount(Context context, String accountName) {
    // 创建账户类型和默认账户
    Account newAccount = new Account(
        accountName, "com.yourpackage");
    // 获取Android账户管理器的实例
    AccountManager accountManager =
        (AccountManager) context.getSystemService(
            ACCOUNT_SERVICE);
    /*
    * 添加账户和账户类型，不设置密码或用户数据
    * 如果成功，返回Account对象，否则报告错误。
    */
    if (accountManager.addAccountExplicitly(newAccount, null, null)) {
        /*
        * 如果你没有在清单文件的<provider>元素中设置android:syncable="true"
        * 则需要调用context.setIsSyncable(account, AUTHORITY,
        * 1)
        */
    }
}

```

```

}

/*
* query() always returns no results
*/
@Override
public Cursor query(
    Uri uri,
    String[] projection,
    String selection,
    String[] selectionArgs,
    String sortOrder) {
    return null;
}

/*
* insert() always returns null (no URI)
*/
@Override
public Uri insert(Uri uri, ContentValues values) {
    return null;
}

/*
* delete() always returns "no rows affected" (0)
*/
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    return 0;
}

/*
* update() always returns "no rows affected" (0)
*/
public int update(
    Uri uri,
    ContentValues values,
    String selection,
    String[] selectionArgs) {
    return 0;
}
}

```

Call this function on successful login to create an account with the logged-in user ID

```

public Account CreateSyncAccount(Context context, String accountName) {
    // Create the account type and default account
    Account newAccount = new Account(
        accountName, "com.yourpackage");
    // Get an instance of the Android account manager
    AccountManager accountManager =
        (AccountManager) context.getSystemService(
            ACCOUNT_SERVICE);
    /*
    * Add the account and account type, no password or user data
    * If successful, return the Account object, otherwise report an error.
    */
    if (accountManager.addAccountExplicitly(newAccount, null, null)) {
        /*
        * If you don't set android:syncable="true" in
        * in your <provider> element in the manifest,
        * then call context.setIsSyncable(account, AUTHORITY, 1)
        */
    }
}

```

```
* 这里。  
    */  
} else {  
    /*  
     * 账户已存在或发生了其他错误。记录日志，报告错误，  
     * 或者内部处理。  
     */  
}  
return newAccount;  
}
```

强制同步

```
Bundle bundle = new Bundle();  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED, true);  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_FORCE, true);  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL, true);  
ContentResolver.requestSync(null, MyContentProvider.getAuthority(), bundle);
```

```
* here.  
*/  
} else {  
    /*  
     * The account exists or some other error occurred. Log this, report it,  
     * or handle it internally.  
    */  
}  
return newAccount;  
}
```

Forcing a Sync

```
Bundle bundle = new Bundle();  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED, true);  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_FORCE, true);  
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL, true);  
ContentResolver.requestSync(null, MyContentProvider.getAuthority(), bundle);
```

第90章：PorterDuff模式

PorterDuff被描述为一种将图像组合在一起的方法，就像将“形状不规则的纸板片”叠加在一起一样，同时也是一种混合重叠部分的方案

第90.1节：创建PorterDuff颜色滤镜

`PorterDuff.Mode`用于创建一个`PorterDuffColorFilter`。颜色滤镜会修改视觉资源中每个像素的颜色。

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

上述滤镜会将非透明像素着色为蓝色。

颜色滤光片可以应用于`Drawable`：

```
drawable.setColorFilter(filter);
```

它可以应用于`ImageView`：

```
imageView.setColorFilter(filter);
```

此外，它也可以应用于`Paint`，这样使用该画笔绘制的颜色会被滤镜修改：

```
paint.setColorFilter(filter);
```

第90.2节：创建PorterDuff XferMode

一个`Xfermode`（可理解为“传输”模式）作为绘制管线中的传输步骤。当`Xfermode`应用于`Paint`时，使用该画笔绘制的像素会根据模式与已绘制的底层像素进行合成：

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

现在我们有了一个蓝色滤镜画笔。任何绘制的形状都会在该形状区域内，将已有的非透明像素染成蓝色。

第90.3节：使用PorterDuffXfermode对位图应用径向遮罩（渐晕）

```
/*
 * 对位图应用径向遮罩（渐晕，即边缘渐变为黑色）
 * @param imageToApplyMaskTo 需要修改的位图
 */
public static void radialMask(Bitmap imageToApplyMaskTo) {
    Canvas canvas = new Canvas(imageToApplyMaskTo);

    final float centerX = imageToApplyMaskTo.getWidth() * 0.5f;
    final float centerY = imageToApplyMaskTo.getHeight() * 0.5f;
    final float radius = imageToApplyMaskTo.getHeight() * 0.7f;

    RadialGradient gradient = new RadialGradient(centerX, centerY, radius,
        0x00000000, 0xFF000000, android.graphics.Shader.TileMode.CLAMP);
```

Chapter 90: PorterDuff Mode

PorterDuff is described as a way of combining images as if they were "irregular shaped pieces of cardboard" overlaid on each other, as well as a scheme for blending the overlapping parts

Section 90.1: Creating a PorterDuff ColorFilter

`PorterDuff.Mode` is used to create a `PorterDuffColorFilter`. A color filter modifies the color of each pixel of a visual resource.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

The above filter will tint the non-transparent pixels to blue color.

The color filter can be applied to a `Drawable`:

```
drawable.setColorFilter(filter);
```

It can be applied to an `ImageView`:

```
imageView.setColorFilter(filter);
```

Also, it can be applied to a `Paint`, so that the color that is drawn using that paint, is modified by the filter:

```
paint.setColorFilter(filter);
```

Section 90.2: Creating a PorterDuff XferMode

An `Xfermode` (think "transfer" mode) works as a transfer step in drawing pipeline. When an `Xfermode` is applied to a `Paint`, the pixels drawn with the paint are combined with underlying pixels (already drawn) as per the mode:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Now we have a blue tint paint. Any shape drawn will tint the already existing, non-transparent pixels blue in the area of the shape.

Section 90.3: Apply a radial mask (vignette) to a bitmap using PorterDuffXfermode

```
/*
 * Apply a radial mask (vignette, i.e. fading to black at the borders) to a bitmap
 * @param imageToApplyMaskTo Bitmap to modify
 */
public static void radialMask(Bitmap imageToApplyMaskTo) {
    Canvas canvas = new Canvas(imageToApplyMaskTo);

    final float centerX = imageToApplyMaskTo.getWidth() * 0.5f;
    final float centerY = imageToApplyMaskTo.getHeight() * 0.5f;
    final float radius = imageToApplyMaskTo.getHeight() * 0.7f;

    RadialGradient gradient = new RadialGradient(centerX, centerY, radius,
        0x00000000, 0xFF000000, android.graphics.Shader.TileMode.CLAMP);
```

```
Paint p = new Paint();
p.setShader(gradient);
p.setColor(0xFF000000);
p.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST_OUT));
    canvas.drawRect(0, 0, imageToApplyMaskTo.getWidth(), imageToApplyMaskTo.getHeight(), p);
}
```

```
Paint p = new Paint();
p.setShader(gradient);
p.setColor(0xFF000000);
p.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST_OUT));
    canvas.drawRect(0, 0, imageToApplyMaskTo.getWidth(), imageToApplyMaskTo.getHeight(), p);
}
```

第91章：菜单

参数	描述
inflate(int menuRes, Menu menu)	从指定的XML资源中填充菜单层级。
getMenuInflater ()	返回一个带有此上下文的MenuInflater。
onCreateOptionsMenu (Menu menu)	初始化Activity标准选项菜单的内容。你应该将菜单项放入menu中。
onOptionsItemSelected (MenuItem item)	每当选中选项菜单中的某个项时都会调用此方法

第91.1节：带分隔符的选项菜单

在安卓中，有一个默认的选项菜单，可以包含多个选项。如果需要显示更多选项，那么将这些选项分组以保持清晰是有意义的。选项可以通过在它们之间放置分隔线（即水平线）来分组。为了允许使用分隔线，可以使用以下主题：

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- 在这里自定义你的主题。 -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:dropDownListViewStyle">@style/PopupMenuListView</item>
</style>
<style name="PopupMenuListView" parent="@style/Widget.AppCompat.ListView.DropDown">
    <item name="android:divider">@color/black</item>
    <item name="android:dividerHeight">1dp</item>
</style>
```

通过更改主题，可以向菜单添加分隔线。

第91.2节：为菜单应用自定义字体

```
public static void applyFontToMenu(Menu m, Context mContext) {
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu()) {
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    }
    Typeface typeface = Typeface.createFromAsset(mContext.getAssets(), "fonts/yourCustomFont.ttf");
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
        Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

然后在 Activity 中：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
}
```

Chapter 91: Menu

Parameter	Description
inflate(int menuRes, Menu menu)	Inflate a menu hierarchy from the specified XML resource.
getMenuInflater ()	Returns a MenuInflater with this context.
onCreateOptionsMenu (Menu menu)	Initialize the contents of the Activity's standard options menu. You should place your menu items in to menu.
onOptionsItemSelected (MenuItem item)	This method is called whenever an item in your options menu is selected

Section 91.1: Options menu with dividers

In Android there is a default options menu, which can take a number of options. If a larger number of options needs to be displayed, then it makes sense to group those options in order to maintain clarity. Options can be grouped by putting dividers (i.e. horizontal lines) between them. In order to allow for dividers, the following theme can be used:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:dropDownListViewStyle">@style/PopupMenuListView</item>
</style>
<style name="PopupMenuListView" parent="@style/Widget.AppCompat.ListView.DropDown">
    <item name="android:divider">@color/black</item>
    <item name="android:dividerHeight">1dp</item>
</style>
```

By changing the theme, dividers can be added to a menu.

Section 91.2: Apply custom font to Menu

```
public static void applyFontToMenu(Menu m, Context mContext) {
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu()) {
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    }
    Typeface font = Typeface.createFromAsset(mContext.getAssets(), "fonts/yourCustomFont.ttf");
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
        Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

and then in the Activity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
}
```

```
    return true;
}
```

第 91.3 节：在 Activity 中创建菜单

要定义您自己的菜单，请在项目的 res/menu/ 目录下创建一个 XML 文件，并使用以下元素构建菜单：

- <menu> : 定义一个菜单，包含所有菜单项。
- <item> : 创建一个菜单项 (MenuItem) ，表示菜单中的单个项目。我们也可以创建嵌套元素以创建子菜单。

步骤 1：

创建如下的 xml 文件：

在 res/menu/main_menu.xml 中：

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/aboutMenu"
        android:title="关于" />
    <item
        android:id="@+id/helpMenu"
        android:title="帮助" />
    <item
        android:id="@+id/signOutMenu"
        android:title="登出" />
</menu>
```

步骤 2：

要指定选项菜单，请在您的活动 (activity) 中重写 onCreateOptionsMenu()。

在此方法中，您可以加载菜单资源（定义在您的XML文件中，即res/menu/main_menu.xml）

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_menu, menu);
    return true;
}
```

当用户从选项菜单中选择一个项目时，系统会调用您activity重写的 onOptionsItemSelected()方法。

- 此方法会传递所选的MenuItem。
- 您可以通过调用getItemId()来识别该项目，该方法返回菜单项的唯一ID（由菜单资源中的android:id属性定义 - res/menu/main_menu.xml）*/

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.aboutMenu:
            Log.d(TAG, "Clicked on About!");
    }
}
```

```
    return true;
}
```

Section 91.3: Creating a Menu in an Activity

To define your own menu, create an XML file inside your project's res/menu/ directory and build the menu with the following elements:

- <menu> : Defines a Menu, which holds all the menu items.
- <item> : Creates a MenuItem, which represents a single item in a menu. We can also create a nested element in order to create a submenu.

Step 1:

Create your own xml file as the following:

In res/menu/main_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/aboutMenu"
        android:title="About" />
    <item
        android:id="@+id/helpMenu"
        android:title="Help" />
    <item
        android:id="@+id/signOutMenu"
        android:title="Sign Out" />
</menu>
```

Step 2:

To specify the options menu, override onCreateOptionsMenu() in your activity.

In this method, you can inflate your menu resource (defined in your XML file i.e., res/menu/main_menu.xml)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_menu, menu);
    return true;
}
```

When the user selects an item from the options menu, the system calls your activity's overridden onOptionsItemSelected() method.

- This method passes the MenuItem selected.
- You can identify the item by calling getItemId(), which returns the unique ID for the menu item (defined by the android:id attribute in the menu resource - res/menu/main_menu.xml)*/

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.aboutMenu:
            Log.d(TAG, "Clicked on About!");
    }
}
```

```

    // 关于的代码写在这里
    return true;
    case R.id.helpMenu:
Log.d(TAG, "点击了帮助 !");
    // 帮助的代码写在这里
    return true;
    case R.id.signOutMenu:
Log.d(TAG, "点击了登出 !");
    // 登出方法调用写在这里
    return true;
default:
    return super.onOptionsItemSelected(item);
}
}

```

收尾工作 !

您的活动代码应如下所示 :

```

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "mytag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.aboutMenu:
Log.d(TAG, "点击了关于 !");
                // 关于的代码写在这里
                return true;
            case R.id.helpMenu:
Log.d(TAG, "点击了帮助 !");
                // 帮助的代码写在这里
                return true;
            case R.id.signOutMenu:
Log.d(TAG, "用户已登出");
                // 登出方法调用写在这里
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}

```

你的菜单外观截图 :

```

    // Code for About goes here
    return true;
    case R.id.helpMenu:
Log.d(TAG, "Clicked on Help !");
    // Code for Help goes here
    return true;
    case R.id.signOutMenu:
Log.d(TAG, "Clicked on Sign Out !");
    // SignOut method call goes here
    return true;
default:
    return super.onOptionsItemSelected(item);
}
}

```

Wrapping up!

Your Activity code should look like below:

```

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "mytag";

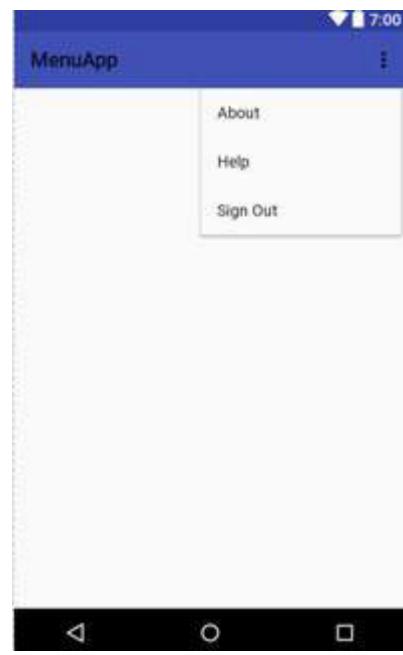
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.aboutMenu:
                Log.d(TAG, "Clicked on About !");
                // Code for About goes here
                return true;
            case R.id.helpMenu:
                Log.d(TAG, "Clicked on Help !");
                // Code for Help goes here
                return true;
            case R.id.signOutMenu:
                Log.d(TAG, "User signed out");
                // SignOut method call goes here
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}

```

Screenshot of how your own Menu looks:



第92章：毕加索

毕加索是一个用于Android的图像库。它由Square创建和维护。它简化了从外部位置显示图像的过程。该库处理从初始HTTP请求到图像缓存的每个阶段。在许多情况下，只需几行代码即可实现这个简洁的库。

第92.1节：将Picasso库添加到您的Android项目中

摘自官方文档：

Gradle.

```
dependencies {  
    compile "com.squareup.picasso:picasso:2.5.2"  
}
```

Maven :

```
<dependency>  
    <groupId>com.squareup.picasso</groupId>  
    <artifactId>picasso</artifactId>  
    <version>2.5.2</version>  
</dependency>
```

第92.2节：使用Picasso实现圆形头像

这是一个基于原始版本的Picasso圆形变换类示例，增加了细边框，并且包含用于堆叠的可选分隔符功能：

```
import android.graphics.Bitmap;  
import android.graphics.BitmapShader;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Paint.Style;  
  
import com.squareup.picasso.Transformation;  
  
public class CircleTransform implements Transformation {  
  
    boolean mCircleSeparator = false;  
  
    public CircleTransform(){  
    }  
  
    public CircleTransform(boolean circleSeparator){  
        mCircleSeparator = circleSeparator;  
    }  
  
    @Override  
    public Bitmap transform(Bitmap source) {  
        int size = Math.min(source.getWidth(), source.getHeight());  
  
        int x = (source.getWidth() - size) / 2;  
        int y = (source.getHeight() - size) / 2;  
  
        Bitmap squaredBitmap = Bitmap.createBitmap(source, x, y, size, size);  
  
        if (squaredBitmap != source) {  
            source.recycle();  
        }  
    }  
}
```

Chapter 92: Picasso

[Picasso](#) is an image library for Android. It's created and maintained by [Square](#). It simplifies the process of displaying images from external locations. The library handles every stage of the process, from the initial HTTP request to the caching of the image. In many cases, only a few lines of code are required to implement this neat library.

Section 92.1: Adding Picasso Library to your Android Project

From the [official documentation](#):

Gradle.

```
dependencies {  
    compile "com.squareup.picasso:picasso:2.5.2"  
}
```

Maven:

```
<dependency>  
    <groupId>com.squareup.picasso</groupId>  
    <artifactId>picasso</artifactId>  
    <version>2.5.2</version>  
</dependency>
```

Section 92.2: Circular Avatars with Picasso

Here is an example Picasso Circle Transform class based on [the original](#), with the addition of a thin border, and also includes functionality for an optional separator for stacking:

```
import android.graphics.Bitmap;  
import android.graphics.BitmapShader;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Paint.Style;  
  
import com.squareup.picasso.Transformation;  
  
public class CircleTransform implements Transformation {  
  
    boolean mCircleSeparator = false;  
  
    public CircleTransform(){  
    }  
  
    public CircleTransform(boolean circleSeparator){  
        mCircleSeparator = circleSeparator;  
    }  
  
    @Override  
    public Bitmap transform(Bitmap source) {  
        int size = Math.min(source.getWidth(), source.getHeight());  
  
        int x = (source.getWidth() - size) / 2;  
        int y = (source.getHeight() - size) / 2;  
  
        Bitmap squaredBitmap = Bitmap.createBitmap(source, x, y, size, size);  
  
        if (squaredBitmap != source) {  
            source.recycle();  
        }  
    }  
}
```

```

}

Bitmap bitmap = Bitmap.createBitmap(size, size, source.getConfig());

    Canvas canvas = new Canvas(bitmap);
BitmapShader shader = new BitmapShader(squaredBitmap, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP);
    Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG | Paint.DITHER_FLAG |
Paint.FILTER_BITMAP_FLAG);
paint.setShader(shader);

    float r = size/2f;
canvas.drawCircle(r, r, r-1, paint);

// 制作细边框：
Paint paintBorder = new Paint();
paintBorder.setStyle(Style.STROKE);
paintBorder.setColor(Color.argb(84, 0, 0, 0));
paintBorder.setAntiAlias(true);
paintBorder.setStrokeWidth(1);
canvas.drawCircle(r, r, r-1, paintBorder);

// 可选的堆叠分隔线：
if (mCircleSeparator) {
    Paint paintBorderSeparator = new Paint();
    paintBorderSeparator.setStyle(Style.STROKE);
    paintBorderSeparator.setColor(Color.parseColor("#ffffffff"));
    paintBorderSeparator.setAntiAlias(true);
paintBorderSeparator.setStrokeWidth(4);
    canvas.drawCircle(r, r, r+1, paintBorderSeparator);
}

squaredBitmap.recycle();
    return bitmap;
}

@Override
public String key() {
    return "circle";
}
}

```

以下是加载图片时的用法（假设this是一个Activity上下文，url是图片URL的字符串）：

```

ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
.fit()
.transform(new CircleTransform())
.into(ivAvatar);

```

结果：

```

}

Bitmap bitmap = Bitmap.createBitmap(size, size, source.getConfig());

    Canvas canvas = new Canvas(bitmap);
BitmapShader shader = new BitmapShader(squaredBitmap, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP);
    Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG | Paint.DITHER_FLAG |
Paint.FILTER_BITMAP_FLAG);
paint.setShader(shader);

    float r = size/2f;
canvas.drawCircle(r, r, r-1, paint);

// Make the thin border:
Paint paintBorder = new Paint();
paintBorder.setStyle(Style.STROKE);
paintBorder.setColor(Color.argb(84, 0, 0, 0));
paintBorder.setAntiAlias(true);
paintBorder.setStrokeWidth(1);
canvas.drawCircle(r, r, r-1, paintBorder);

// Optional separator for stacking:
if (mCircleSeparator) {
    Paint paintBorderSeparator = new Paint();
    paintBorderSeparator.setStyle(Style.STROKE);
    paintBorderSeparator.setColor(Color.parseColor("#ffffffff"));
    paintBorderSeparator.setAntiAlias(true);
    paintBorderSeparator.setStrokeWidth(4);
    canvas.drawCircle(r, r, r+1, paintBorderSeparator);
}

squaredBitmap.recycle();
    return bitmap;
}

@Override
public String key() {
    return "circle";
}
}

```

Here is how to use it when loading an image (assuming **this** is an Activity Context, and url is a String with the url of the image to load):

```

ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
.fit()
.transform(new CircleTransform())
.into(ivAvatar);

```

Result:



testy_s3

Nexus 6



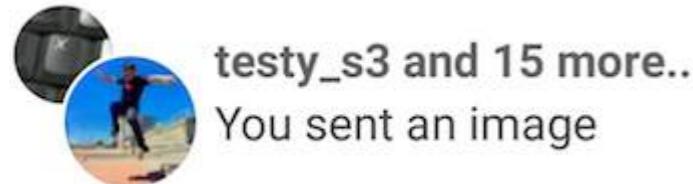
testy_ver222

Testy 222

对于带分隔符的用法，给顶部图片的构造函数传入true：

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
.fit()
.transform(new CircleTransform(true))
.into(ivAvatar);
```

结果 (FrameLayout中的两个ImageView)：



第92.3节：占位符和错误处理

Picasso支持下载占位符和错误占位符作为可选功能。它还提供了回调来处理下载结果。

```
Picasso.with(context)
.load("YOUR IMAGE URL HERE")
.placeholder(Your Drawable Resource) //这是可选的，表示在URL图片下载时显示的图片
.error(Your Drawable Resource) //这也是可选的，如果下载图片时发生错误，将显示此图片
.into(imageView, new Callback(){
    @Override
    public void onSuccess() {}

    @Override
    public void onError() {}
});
```

请求将在显示错误占位符之前重试三次。

第92.4节：调整大小和旋转

```
Picasso.with(context)
.load("YOUR IMAGE URL HERE")
.placeholder(DRAWABLE RESOURCE) // 可选
.error(DRAWABLE RESOURCE) // 可选
.resize(width, height) // 可选
.rotate(degree) // 可选
.into(imageView);
```



testy_s3

Nexus 6



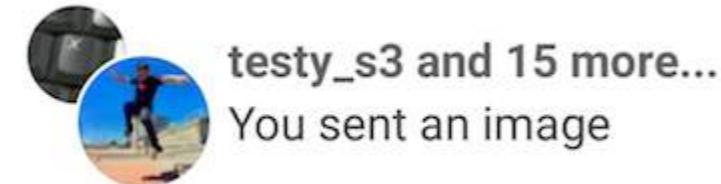
testy_ver222

Testy 222

For use with the separator, give **true** to the constructor for the top image:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
.fit()
.transform(new CircleTransform(true))
.into(ivAvatar);
```

Result (two ImageViews in a FrameLayout):



Section 92.3: Placeholder and Error Handling

Picasso supports both download and error placeholders as optional features. It also provides callbacks for handling the download result.

```
Picasso.with(context)
.load("YOUR IMAGE URL HERE")
.placeholder(Your Drawable Resource) //this is optional the image to display while the url image
is downloading
.error(Your Drawable Resource) //this is also optional if some error has occurred in
downloading the image this image would be displayed
.into(imageView, new Callback(){
    @Override
    public void onSuccess() {}

    @Override
    public void onError() {}
});
```

A request will be retried three times before the error placeholder is shown.

Section 92.4: Re-sizing and Rotating

```
Picasso.with(context)
.load("YOUR IMAGE URL HERE")
.placeholder(DRAWABLE RESOURCE) // optional
.error(DRAWABLE RESOURCE) // optional
.resize(width, height) // optional
.rotate(degree) // optional
.into(imageView);
```

第92.5节：在Picasso中禁用缓存

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

第92.6节：将Picasso用作Html.fromHtml的ImageGetter

将Picasso用作ImageGetter用于Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {
    private TextView textView;
    private Picasso picasso;

    public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
        this.picasso = picasso;
        this.textView = textView;
    }

    @Override
    public Drawable getDrawable(String source) {
        Log.d(PicassoImageGetter.class.getName(), "开始加载网址 " + source);
        BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();

        picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

        return drawable;
    }

    private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {
        protected Drawable drawable;

        @Override
        public void draw(Canvas canvas) {
            if (drawable != null) {
                checkBounds();
                drawable.draw(canvas);
            }
        }

        public void setDrawable(@Nullable Drawable drawable) {
            if (drawable != null) {
                this.drawable = drawable;
                checkBounds();
            }
        }

        private void checkBounds() {
            float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
                drawable.getIntrinsicHeight();
        }
    }
}
```

Section 92.5: Disable cache in Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Section 92.6: Using Picasso as ImageGetter for Html.fromHtml

Using Picasso as ImageGetter for Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {
    private TextView textView;
    private Picasso picasso;

    public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
        this.picasso = picasso;
        this.textView = textView;
    }

    @Override
    public Drawable getDrawable(String source) {
        Log.d(PicassoImageGetter.class.getName(), "Start loading url " + source);
        BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();

        picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

        return drawable;
    }

    private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {
        protected Drawable drawable;

        @Override
        public void draw(Canvas canvas) {
            if (drawable != null) {
                checkBounds();
                drawable.draw(canvas);
            }
        }

        public void setDrawable(@Nullable Drawable drawable) {
            if (drawable != null) {
                this.drawable = drawable;
                checkBounds();
            }
        }

        private void checkBounds() {
            float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
                drawable.getIntrinsicHeight();
        }
    }
}
```

```

int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
int height = (int) ((float) width / defaultProportion);

if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

setBounds(0, 0, textView.getWidth(), height); //设置为全宽

    int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
    int halfOfImageWidth = (int) ((float) width / 2f);

drawable.setBounds(
halfOfPlaceHolderWidth - halfOfImageWidth, //使图像居中
0,
halfOfPlaceHolderWidth + halfOfImageWidth,
height);

textView.setText(textView.getText()); //刷新文本
}

//-----//


@Override
public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
    setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
}

@Override
public void onBitmapFailed(Drawable errorDrawable) {
    setDrawable(errorDrawable);
}

@Override
public void onPrepareLoad(Drawable placeHolderDrawable) {
    setDrawable(placeHolderDrawable);
}

//-----//

```

}) 用法很简单：从 TextView 的构造函数中调用 Html.fromHtml 方法即可。

第92.7节：使用Picasso取消图片请求

在某些情况下，我们需要在Picasso的图片下载完成之前取消该下载请求。

这可能由于各种原因发生，例如父视图在图片下载完成之前切换到了其他视图。

在这种情况下，你可以使用cancelRequest()方法取消图片下载请求：

```

ImageView imageView;
//.....

```

```

int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
int height = (int) ((float) width / defaultProportion);

if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

setBounds(0, 0, textView.getWidth(), height); //set to full width

    int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
    int halfOfImageWidth = (int) ((float) width / 2f);

drawable.setBounds(
    halfOfPlaceHolderWidth - halfOfImageWidth, //centering an image
    0,
    halfOfPlaceHolderWidth + halfOfImageWidth,
    height);

    textView.setText(textView.getText()); //refresh text
}
}

//-----//


@Override
public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
    setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
}

@Override
public void onBitmapFailed(Drawable errorDrawable) {
    setDrawable(errorDrawable);
}

@Override
public void onPrepareLoad(Drawable placeHolderDrawable) {
    setDrawable(placeHolderDrawable);
}

//-----//

```

The usage is simple:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Section 92.7: Cancelling Image Requests using Picasso

In certain cases we need to cancel an image download request in Picasso before the download has completed.

This could happen for various reasons, for example if the parent view transitioned to some other view before the image download could be completed.

In this case, you can cancel the image download request using the cancelRequest() method:

```

ImageView imageView;
//.....

```

```
Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

第92.8节：从外部存储加载图像

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
.load(new File(imagePath))
.into(imageView);
```

第92.9节：使用Picasso下载图像为Bitmap

如果你想使用Picasso下载图像为Bitmap，以下代码将对你有帮助：

```
Picasso.with(mContext)
.load(ImageUrl)
进入(新的目标) {
    @Override
    public void onBitmapLoaded(Bitmap 位图, Picasso.LoadedFrom 来源) {
        // 待办事项：在这里对你的位图进行处理
    }

    @Override
    public void onBitmapFailed(Drawable 错误图像) {
    }

    @Override
    public void onPrepareLoad(Drawable 占位图) {
    }
});
```

第92.10节：先尝试离线磁盘缓存，然后在线获取图像

首先将 OkHttp 添加到应用模块的 gradle 构建文件中

```
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.squareup.okhttp:okhttp:2.4.0'
compile 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.0.2'
```

然后创建一个继承自 Application 的类

```
import android.app.Application;

import com.squareup.picasso.OkHttpDownloader;
import com.squareup.picasso.Picasso;

public class Global extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        Picasso.Builder builder = new Picasso.Builder(this);
        builder.downloader(new OkHttpDownloader(this, Integer.MAX_VALUE));
        Picasso built = builder.build();
        built.setIndicatorsEnabled(true);
        built.setLoggingEnabled(true);
    }
}
```

```
Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Section 92.8: Loading Image from external Storage

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
.load(new File(imagePath))
.into(imageView);
```

Section 92.9: Downloading image as Bitmap using Picasso

If you want to Download image as Bitmap using Picasso following code will help you:

```
Picasso.with(mContext)
.load(ImageUrl)
.into(new Target() {
    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
        // Todo: Do something with your bitmap here
    }

    @Override
    public void onBitmapFailed(Drawable errorDrawable) {
    }

    @Override
    public void onPrepareLoad(Drawable placeHolderDrawable) {
    }
});
```

Section 92.10: Try offline disk cache first, then go online and fetch the image

first add the OkHttp to the gradle build file of the app module

```
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.squareup.okhttp:okhttp:2.4.0'
compile 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.0.2'
```

Then make a class extending Application

```
import android.app.Application;

import com.squareup.picasso.OkHttpDownloader;
import com.squareup.picasso.Picasso;

public class Global extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        Picasso.Builder builder = new Picasso.Builder(this);
        builder.downloader(new OkHttpDownloader(this, Integer.MAX_VALUE));
        Picasso built = builder.build();
        built.setIndicatorsEnabled(true);
        built.setLoggingEnabled(true);
    }
}
```

```
Picasso.setSingletonInstance(built);  
}  
}
```

将其添加到Manifest文件中，如下所示：

```
<application  
    android:name=".Global"  
    ... >
```

正常用法

```
Picasso.with(getActivity())  
.load(imageUrl)  
.networkPolicy(NetworkPolicy.OFFLINE)  
.into(imageView, new Callback() {  
    @Override  
    public void onSuccess() {  
        //离线缓存命中  
    }  
  
    @Override  
    public void onError() {  
        //如果缓存失败则尝试在线加载  
        Picasso.with(getActivity())  
            .load(imageUrl)  
            .error(R.drawable.header)  
            .into(imageView, new Callback() {  
                @Override  
                public void onSuccess() {  
                    //在线下载  
                }  
  
                @Override  
                public void onError() {  
                    Log.v("Picasso", "无法获取图片");  
                }  
            });  
    }  
});
```

```
Picasso.setSingletonInstance(built);  
}  
}
```

将其添加到Manifest文件中，如下所示：

```
<application  
    android:name=".Global"  
    ... >  
  
</application>
```

Normal Usage

```
Picasso.with(getActivity())  
.load(imageUrl)  
.networkPolicy(NetworkPolicy.OFFLINE)  
.into(imageView, new Callback() {  
    @Override  
    public void onSuccess() {  
        //Offline Cache hit  
    }  
  
    @Override  
    public void onError() {  
        //Try again online if cache failed  
        Picasso.with(getActivity())  
            .load(imageUrl)  
            .error(R.drawable.header)  
            .into(imageView, new Callback() {  
                @Override  
                public void onSuccess() {  
                    //Online download  
                }  
  
                @Override  
                public void onError() {  
                    Log.v("Picasso", "Could not fetch image");  
                }  
            });  
    }  
});
```

[原答案链接](#)

[Link to original answer](#)

第93章：RoboGuice

第93.1节：简单示例

RoboGuice是一个框架，利用谷歌自己的Guice库，将依赖注入的简洁性和便利性带到Android平台。

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name) TextView name;
    @InjectView(R.id.thumbnail) ImageView thumbnail;
    @InjectResource(R.drawable.icon) Drawable icon;
    @InjectResource(R.string.app_name) String myName;
    @Inject LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText("Hello, " + myName );
    }
}
```

第93.2节：Gradle项目的安装

将以下pom添加到你的gradle构建文件的依赖部分：

```
project.dependencies {
    compile 'org.robolectric:robolectric:3.+'
    provided 'org.robolectric:roboblender:3.+'
}
```

第93.3节：@ContentView注解

@ContentView注解可以进一步简化活动的开发，替代
setContentView语句：

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        textView.setText("Hello!");
    }
}
```

第93.4节：@InjectResource 注解

您可以注入任何类型的资源，字符串、动画、可绘制资源等。

要将第一个资源注入到活动中，您需要：

- 继承自 RoboActivity
- 使用 @InjectResource 注解您的资源

示例

Chapter 93: RoboGuice

Section 93.1: Simple example

RoboGuice is a framework that brings the simplicity and ease of Dependency Injection to Android, using Google's own Guice library.

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name) TextView name;
    @InjectView(R.id.thumbnail) ImageView thumbnail;
    @InjectResource(R.drawable.icon) Drawable icon;
    @InjectResource(R.string.app_name) String myName;
    @Inject LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText("Hello, " + myName );
    }
}
```

Section 93.2: Installation for Gradle Projects

Add the following pom to the dependencies section of your gradle build file :

```
project.dependencies {
    compile 'org.robolectric:roboelectric:3.+'
    provided 'org.robolectric:roboblender:3.+'
}
```

Section 93.3: @ContentView annotation

The @ContentView annotation can be used to further alleviate development of activities and replace the setContentView statement :

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        textView.setText("Hello!");
    }
}
```

Section 93.4: @InjectResource annotation

You can inject any type of resource, Strings, Animations, Drawables, etc.

To inject your first resource into an activity, you'll need to:

- Inherit from RoboActivity
- Annotate your resources with @InjectResource

Example

```
@InjectResource(R.string.app_name) String name;  
  
@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;  
  
@InjectResource(R.anim.my_animation) Animation myAnimation;
```

第93.5节：@InjectView 注解

您可以使用 @InjectView 注解注入任何视图：

你需要：

- 继承自 RoboActivity
- 设置你的内容视图
- 用 @InjectView 注解你的视图

示例

```
@InjectView(R.id.textView1) TextView textView1;  
  
@InjectView(R.id.textView2) TextView textView2;  
  
@InjectView(R.id.imageView1) ImageView imageView1;
```

第 93.6 节：RoboGuice 简介

RoboGuice 是一个框架，利用谷歌自己的 Guice 库，将依赖注入的简洁和便利带到 Android 平台。

RoboGuice 3 精简了你的应用代码。代码越少，出错的机会就越少。它还使你的代码更易于理解——你的代码不再充斥着 Android 平台的机制，而是可以专注于应用独有的实际逻辑。

为了让你了解，看看这个典型 Android Activity 的简单示例：

```
class AndroidWay extends Activity {  
    TextView name;  
    ImageView thumbnail;  
    LocationManager loc;  
    Drawable icon;  
    String myName;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        name = (TextView) findViewById(R.id.name);  
        thumbnail = (ImageView) findViewById(R.id.thumbnail);  
        loc = (LocationManager) getSystemService(Activity.LOCATION_SERVICE);  
        icon = getResources().getDrawable(R.drawable.icon);  
        myName = getString(R.string.app_name);  
        name.setText("Hello, " + myName);  
    }  
}
```

这个示例有19行代码。如果你试图阅读 onCreate()，你必须跳过5行模板初始化代码，才能找到唯一真正重要的行：name.setText()。复杂的活动可能会有更多类似的初始化代码。

```
@InjectResource(R.string.app_name) String name;  
  
@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;  
  
@InjectResource(R.anim.my_animation) Animation myAnimation;
```

Section 93.5: @InjectView annotation

You can inject any view using the @InjectView annotation:

You'll need to:

- Inherit from RoboActivity
- Set your content view
- Annotate your views with @InjectView

Example

```
@InjectView(R.id.textView1) TextView textView1;  
  
@InjectView(R.id.textView2) TextView textView2;  
  
@InjectView(R.id.imageView1) ImageView imageView1;
```

Section 93.6: Introduction to RoboGuice

RoboGuice is a framework that brings the simplicity and ease of Dependency Injection to Android, using Google's own Guice library.

RoboGuice 3 slims down your application code. Less code means fewer opportunities for bugs. It also makes your code easier to follow -- no longer is your code littered with the mechanics of the Android platform, but now it can focus on the actual logic unique to your application.

To give you an idea, take a look at this simple example of a typical Android Activity:

```
class AndroidWay extends Activity {  
    TextView name;  
    ImageView thumbnail;  
    LocationManager loc;  
    Drawable icon;  
    String myName;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        name = (TextView) findViewById(R.id.name);  
        thumbnail = (ImageView) findViewById(R.id.thumbnail);  
        loc = (LocationManager) getSystemService(Activity.LOCATION_SERVICE);  
        icon = getResources().getDrawable(R.drawable.icon);  
        myName = getString(R.string.app_name);  
        name.setText("Hello, " + myName);  
    }  
}
```

This example is 19 lines of code. If you're trying to read through onCreate(), you have to skip over 5 lines of boilerplate initialization to find the only one that really matters: name.setText(). And complex activities can end up with a lot more of this sort of initialization code.

将此与使用 RoboGuice 编写的相同应用进行比较：

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name) TextView name;
    @InjectView(R.id.thumbnail) ImageView thumbnail;
    @InjectResource(R.drawable.icon) Drawable icon;
    @InjectResource(R.string.app_name) String myName;
    @Inject LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText("Hello, " + myName );
    }
}
```

RoboGuice 的目标是让你的代码专注于你的应用，而不是专注于你通常需要在 Android 中维护的所有初始化和生命周期代码。

注释：

@ContentView 注解：

@ContentView注解可以进一步简化活动的开发，替代
setContentView语句：

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        textView.setText("Hello!");
    }
}
```

@InjectResource 注解：

首先你需要一个继承自 RoboActivity 的 Activity。然后，假设你在 res/anim 文件夹中有一个动画文件 my_animation.xml，现在你可以通过注解来引用它：

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // 你的其余代码
}
```

@Inject 注解：

你需要确保你的 Activity 继承自 RoboActivity，并用 @Inject 注解你的系统服务成员变量。
Roboguice 会处理剩下的部分。

```
class MyActivity extends RoboActivity {
    @Inject 震动器 vibrator;
    @Inject 通知管理器 notificationManager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

Compare this to the same app, written using RoboGuice:

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name) TextView name;
    @InjectView(R.id.thumbnail) ImageView thumbnail;
    @InjectResource(R.drawable.icon) Drawable icon;
    @InjectResource(R.string.app_name) String myName;
    @Inject LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText("Hello, " + myName );
    }
}
```

RoboGuice's goal is to make your code be about your app, rather than be about all the initialization and lifecycle code you typically have to maintain in Android.

Annotations:

@ContentView annotation:

The @ContentView annotation can be used to further alleviate development of activities and replace the setContentView statement :

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        textView.setText("Hello!");
    }
}
```

@InjectResource annotation:

First you need an Activity that inherits from RoboActivity. Then, assuming that you have an animation my_animation.xml in your res/anim folder, you can now reference it with an annotation:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@Inject annotation:

You make sure your activity extends from RoboActivity and annotate your System service member with @Inject.
Roboguice will do the rest.

```
class MyActivity extends RoboActivity {
    @Inject Vibrator vibrator;
    @Inject NotificationManager notificationManager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

```
// 我们可以直接使用这些实例！  
vibrator.震动(1000L); // RoboGuice 负责处理 getSystemService(VIBRATOR_SERVICE)  
notificationManager.取消所有();
```

除了视图、资源、服务和其他 Android 特有的内容外，RoboGuice 还可以注入普通的 Java 对象（POJO）。默认情况下，RoboGuice 会调用你的 POJO 的无参构造函数

```
class MyActivity extends RoboActivity {  
    @Inject Foo foo; // 这基本上会调用 new Foo();  
}
```

```
// we can use the instances directly!  
vibrator.vibrate(1000L); // RoboGuice took care of the getSystemService(VIBRATOR_SERVICE)  
notificationManager.cancelAll();
```

In addition to Views, Resources, Services, and other android-specific things, RoboGuice can inject Plain Old Java Objects. By default Roboguice will call a no argument constructor on your POJO

```
class MyActivity extends RoboActivity {  
    @Inject Foo foo; // this will basically call new Foo();  
}
```

第94章：ACRA

参数	描述
@ReportCrashes	定义了 ACRA 的设置，例如报告位置、自定义内容等
formUri	报告崩溃的文件路径

第94.1节：ACRA处理器

示例应用程序-扩展类用于处理报告：

```
@ReportsCrashes(  
  
    formUri = "https://backend-of-your-choice.com/", //无密码保护。  
    customReportContent = { /* */ReportField.APP_VERSION_NAME,  
    ReportField.PACKAGE_NAME,ReportField.ANDROID_VERSION, ReportField.PHONE_MODEL,ReportField.LOGCAT },  
    mode = ReportingInteractionMode.TOAST,  
    resToastText = R.string.crash  
  
)  
public class ACRAHandler extends Application {  
    @Override  
    protected void attachBaseContext(Context base) {  
        super.attachBaseContext(base);  
  
        final ACRAConfiguration config = new ConfigurationBuilder(this)  
  
.build();  
  
        // 初始化ACRA  
        ACRA.init(this, config);  
  
    }  
}
```

第94.2节：示例清单

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    <!-- 等等 -->  
  
>  
  
<!-- 需要互联网。READ_LOGS权限用于确保Logcat日志被传输 --&gt;<br/><uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.READ_LOGS"/>  
  
<application  
    android:allowBackup="true"  
    android:name=".ACRAHandler"<!-- 启动时激活ACRA -->  
        android:icon="@drawable/ic_launcher"  
        android:label="@string/app_name"  
        android:theme="@style/AppTheme" >
```

Chapter 94: ACRA

Parameter	Description
@ReportCrashes	Defines the ACRA settings such as where it is to be reported, custom content, etc
formUri	the path to the file that reports the crash

Section 94.1: ACRAHandler

Example Application-extending class for handling the reporting:

```
@ReportsCrashes(  
  
    formUri = "https://backend-of-your-choice.com/", //Non-password protected.  
    customReportContent = { /* */ReportField.APP_VERSION_NAME,  
    ReportField.PACKAGE_NAME,ReportField.ANDROID_VERSION, ReportField.PHONE_MODEL,ReportField.LOGCAT },  
    mode = ReportingInteractionMode.TOAST,  
    resToastText = R.string.crash  
  
)  
public class ACRAHandler extends Application {  
    @Override  
    protected void attachBaseContext(Context base) {  
        super.attachBaseContext(base);  
  
        final ACRAConfiguration config = new ConfigurationBuilder(this)  
  
.build();  
  
        // Initialise ACRA  
        ACRA.init(this, config);  
  
    }  
}
```

Section 94.2: Example manifest

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    <!-- etc -->  
  
>  
  
<!-- Internet is required. READ_LOGS are to ensure that the Logcat is transmitted--&gt;<br/><uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.READ_LOGS"/>  
  
<application  
    android:allowBackup="true"  
    android:name=".ACRAHandler"<!-- Activates ACRA on startup -->  
        android:icon="@drawable/ic_launcher"  
        android:label="@string/app_name"  
        android:theme="@style/AppTheme" >
```

```
<!-- 活动 -->
```

```
</manifest>
```

第94.3节：安装

Maven

```
<dependency>
    <groupId>ch.acra</groupId>
    <artifactId>acra</artifactId>
    <version>4.9.2</version>
    <type>aar</type>
</dependency>
```

Gradle

```
compile 'ch.acra:acra:4.9.2'
```

```
<!-- Activities -->
```

```
</application>
```

```
</manifest>
```

Section 94.3: Installation

Maven

```
<dependency>
    <groupId>ch.acra</groupId>
    <artifactId>acra</artifactId>
    <version>4.9.2</version>
    <type>aar</type>
</dependency>
```

Gradle

```
compile 'ch.acra:acra:4.9.2'
```

第95章：Parcelable

Parcelable是Android特有的接口，您需要自己实现序列化。它被创建得比Serializable高效得多，且能解决默认Java序列化方案的一些问题。

第95.1节：制作自定义的Parcelable对象

```
/**  
 * 由亚历克斯·沙利文于2016年7月21日创建。  
 */  
public class Foo implements Parcelable  
{  
    private final int myFirstVariable;  
    private final String mySecondVariable;  
    private final long myThirdVariable;  
  
    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)  
    {  
        this.myFirstVariable = myFirstVariable;  
        this.mySecondVariable = mySecondVariable;  
        this.myThirdVariable = myThirdVariable;  
    }  
  
    // 注意，您必须按照写入parcel的顺序读取parcel中的值！此方法是我们自定义的方法// 用于  
    // 从Parcel实例化对象。它在下面声明的Parcelable.Creator变量中使用。  
  
    public Foo(Parcel in)  
    {  
        this.myFirstVariable = in.readInt();  
        this.mySecondVariable = in.readString();  
        this.myThirdVariable = in.readLong();  
    }  
  
    // describeContents方法通常返回0。当被打包的对象包含文件描述符时使用。  
  
    @Override  
    public int describeContents()  
    {  
        return 0;  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags)  
    {  
        dest.writeInt(myFirstVariable);  
        dest.writeString(mySecondVariable);  
        dest.writeLong(myThirdVariable);  
    }  
  
    // 请注意，这个看似随机的字段并非可选。系统会通过反射查找此变量，以便在从Intent读取时  
    // 实例化你的打包对象。  
  
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()  
    {  
        // 此方法用于实际实例化我们的自定义对象  
        // 从Parcel中。惯例是我们创建一个新的构造函数，  
        // 该构造函数以parcel作为唯一参数。  
        public Foo createFromParcel(Parcel in)  
        {  
    }
```

Chapter 95: Parcelable

Parcelable is an Android specific interface where you implement the serialization yourself. It was created to be far more efficient than Serializable, and to get around some problems with the default Java serialization scheme.

Section 95.1: Making a custom object Parcelable

```
/**  
 * Created by Alex Sullivan on 7/21/16.  
 */  
public class Foo implements Parcelable  
{  
    private final int myFirstVariable;  
    private final String mySecondVariable;  
    private final long myThirdVariable;  
  
    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)  
    {  
        this.myFirstVariable = myFirstVariable;  
        this.mySecondVariable = mySecondVariable;  
        this.myThirdVariable = myThirdVariable;  
    }  
  
    // Note that you MUST read values from the parcel IN THE SAME ORDER that  
    // values were WRITTEN to the parcel! This method is our own custom method  
    // to instantiate our object from a Parcel. It is used in the Parcelable.Creator variable we  
    // declare below.  
    public Foo(Parcel in)  
    {  
        this.myFirstVariable = in.readInt();  
        this.mySecondVariable = in.readString();  
        this.myThirdVariable = in.readLong();  
    }  
  
    // The describeContents method can normally return 0. It's used when  
    // the parcelled object includes a file descriptor.  
    @Override  
    public int describeContents()  
    {  
        return 0;  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags)  
    {  
        dest.writeInt(myFirstVariable);  
        dest.writeString(mySecondVariable);  
        dest.writeLong(myThirdVariable);  
    }  
  
    // Note that this seemingly random field IS NOT OPTIONAL. The system will  
    // look for this variable using reflection in order to instantiate your  
    // parcelled object when read from an Intent.  
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()  
    {  
        // This method is used to actually instantiate our custom object  
        // from the Parcel. Convention dictates we make a new constructor that  
        // takes the parcel in as its only argument.  
        public Foo createFromParcel(Parcel in)  
        {  
    }
```

```

        return new Foo(in);
    }

    // 此方法用于创建自定义对象的数组。
    // 通常声明一个指定大小的新数组就足够了。
    public Foo[] newArray(int size)
    {
        return new Foo[size];
    }
}

```

第95.2节：包含另一个 Parcelable 对象的 Parcelable 对象

包含一个可打包类的示例：

```

public class Repository implements Parcelable {
    private String name;
    private Owner owner;
    private boolean isPrivate;

    public Repository(String name, Owner owner, boolean isPrivate) {
        this.name = name;
        this.owner = owner;
        this.isPrivate = isPrivate;
    }

    protected Repository(Parcel in) {
        name = in.readString();
        owner = in.readParcelable(Owner.class.getClassLoader());
        isPrivate = in.readByte() != 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeParcelable(owner, flags);
        dest.writeByte((byte) (isPrivate ? 1 : 0));
    }

    @Override
    public int describeContents() {
        return 0;
    }

    public static final Creator<Repository> CREATOR = new Creator<Repository>() {
        @Override
        public Repository createFromParcel(Parcel in) {
            return new Repository(in);
        }

        @Override
        public Repository[] newArray(int size) {
            return new Repository[size];
        }
    };

    // 获取器和设置器

    public String getName() {

```

```

        return new Foo(in);
    }

    // This method is used to make an array of your custom object.
    // Declaring a new array with the provided size is usually enough.
    public Foo[] newArray(int size)
    {
        return new Foo[size];
    }
}

```

Section 95.2: Parcelable object containing another Parcelable object

An example of a class that contains a parcelable class inside:

```

public class Repository implements Parcelable {
    private String name;
    private Owner owner;
    private boolean isPrivate;

    public Repository(String name, Owner owner, boolean isPrivate) {
        this.name = name;
        this.owner = owner;
        this.isPrivate = isPrivate;
    }

    protected Repository(Parcel in) {
        name = in.readString();
        owner = in.readParcelable(Owner.class.getClassLoader());
        isPrivate = in.readByte() != 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeParcelable(owner, flags);
        dest.writeByte((byte) (isPrivate ? 1 : 0));
    }

    @Override
    public int describeContents() {
        return 0;
    }

    public static final Creator<Repository> CREATOR = new Creator<Repository>() {
        @Override
        public Repository createFromParcel(Parcel in) {
            return new Repository(in);
        }

        @Override
        public Repository[] newArray(int size) {
            return new Repository[size];
        }
    };

    // getters and setters

    public String getName() {

```

```

    return name;
}

public void setName(String name) {
    this.name = name;
}

public Owner getOwner() {
    return owner;
}

public void setOwner(Owner owner) {
    this.owner = owner;
}

public boolean isPrivate() {
    return isPrivate;
}

public void setPrivate(boolean isPrivate) {
    this.isPrivate = isPrivate;
}
}

```

Owner 只是一个普通的可打包类 (parcelable class)。

第95.3节：使用枚举（Enums）与 Parcelable

```

/**
 * 由 Nick Cardoso 于 2016年03月08日 创建。
 * 这不是一个完整的 Parcelable 实现，它仅突出展示了读取和写入枚举值到包裹 (parcel) 的最简单方法
 */
public class Foo implements Parcelable {

    private final MyEnum myEnumVariable;
    private final MyEnum mySaferEnumVariableExample;

    public Foo(Parcel in) {

        //最简单的方法
        myEnumVariable = MyEnum.valueOf( in.readString() );

        //带有一些错误检查
        try {
            mySaferEnumVariableExample= MyEnum.valueOf( in.readString() );
        } catch (IllegalArgumentException e) { //错误的字符串或空值
            mySaferEnumVariableExample= MyEnum.DEFAULT;
        }
    }

    ...

    @Override
    public void writeToParcel(Parcel dest, int flags) {

        //简单的方法
        dest.writeString(myEnumVariable.name());

        //通过一些错误检查避免空指针异常
    }
}

```

```

    return name;
}

public void setName(String name) {
    this.name = name;
}

public Owner getOwner() {
    return owner;
}

public void setOwner(Owner owner) {
    this.owner = owner;
}

public boolean isPrivate() {
    return isPrivate;
}

public void setPrivate(boolean isPrivate) {
    this.isPrivate = isPrivate;
}
}

```

Owner is just a normal parcelable class.

Section 95.3: Using Enums with Parcelable

```

/**
 * Created by Nick Cardoso on 03/08/16.
 * This is not a complete parcelable implementation, it only highlights the easiest
 * way to read and write your Enum values to your parcel
 */
public class Foo implements Parcelable {

    private final MyEnum myEnumVariable;
    private final MyEnum mySaferEnumVariableExample;

    public Foo(Parcel in) {

        //the simplest way
        myEnumVariable = MyEnum.valueOf( in.readString() );

        //with some error checking
        try {
            mySaferEnumVariableExample= MyEnum.valueOf( in.readString() );
        } catch (IllegalArgumentException e) { //bad string or null value
            mySaferEnumVariableExample= MyEnum.DEFAULT;
        }
    }

    ...

    @Override
    public void writeToParcel(Parcel dest, int flags) {

        //the simple way
        dest.writeString(myEnumVariable.name());

        //avoiding NPEs with some error checking
    }
}

```

```
dest.writeString(mySaferEnumVariableExample == null? null :  
mySaferEnumVariableExample.name());  
  
}  
  
}  
  
public enum MyEnum {  
    VALUE_1,  
    VALUE_2,  
    DEFAULT  
}
```

这比（例如）使用序数更可取，因为向枚举中插入新值不会影响之前存储的值

```
dest.writeString(mySaferEnumVariableExample == null? null :  
mySaferEnumVariableExample.name());  
  
}  
  
}  
  
public enum MyEnum {  
    VALUE_1,  
    VALUE_2,  
    DEFAULT  
}
```

This is preferable to (for example) using an ordinal, because inserting new values into your enum will not affect previously stored values

第96章：Retrofit2

官方的Retrofit页面自我描述为

一个适用于Android和Java的类型安全REST客户端。

Retrofit将你的REST API转换成Java接口。它使用注解来描述HTTP请求，默认集成了URL参数替换和查询参数支持。此外，它还提供了多部分请求体和文件上传的功能。

第96.1节：一个简单的GET请求

我们将展示如何向一个API发起GET请求，该API响应一个JSON对象或一个JSON数组。

我们首先需要做的是将Retrofit和GSON转换器依赖添加到模块的gradle文件中。

按照备注部分的说明添加retrofit库的依赖。

预期的 JSON 对象示例：

```
{  
    "deviceId": "56V56C14SF5B4SF",  
    "name": "Steven",  
    "eventCount": 0  
}
```

JSON数组示例：

```
[  
    {  
        "deviceId": "56V56C14SF5B4SF",  
        "name": "Steven",  
        "eventCount": 0  
    },  
    {  
        "deviceId": "35A80SF3QDV7M9F",  
        "name": "John",  
        "eventCount": 2  
    }  
]
```

对应的模型类示例：

```
public class Device  
{  
    @SerializedName("deviceId")  
    public String id;  
  
    @SerializedName("name")  
    public String name;  
  
    @SerializedName("eventCount")  
    public int eventCount;  
}
```

这里的@SerializedName注解来自GSON库，允许我们使用序列化名称作为键，将此类序列化和反序列化为JSON格式。现在我们可以构建实际从服务器获取数据的API接口。

Chapter 96: Retrofit2

The official Retrofit page describes itself as

A type-safe REST client for Android and Java.

Retrofit turns your REST API into a Java interface. It uses annotations to describe HTTP requests, URL parameter replacement and query parameter support is integrated by default. Additionally, it provides functionality for multipart request body and file uploads.

Section 96.1: A Simple GET Request

We are going to be showing how to make a GET request to an API that responds with a JSON object or a JSON array. The first thing we need to do is add the Retrofit and GSON Converter dependencies to our module's gradle file.

Add the dependencies for retrofit library as described in the Remarks section.

Example of expected JSON object:

```
{  
    "deviceId": "56V56C14SF5B4SF",  
    "name": "Steven",  
    "eventCount": 0  
}
```

Example of JSON array:

```
[  
    {  
        "deviceId": "56V56C14SF5B4SF",  
        "name": "Steven",  
        "eventCount": 0  
    },  
    {  
        "deviceId": "35A80SF3QDV7M9F",  
        "name": "John",  
        "eventCount": 2  
    }  
]
```

Example of corresponding model class:

```
public class Device  
{  
    @SerializedName("deviceId")  
    public String id;  
  
    @SerializedName("name")  
    public String name;  
  
    @SerializedName("eventCount")  
    public int eventCount;  
}
```

The @SerializedName annotations here are from the GSON library and allows us to serialize and deserialize this class to JSON using the serialized name as the keys. Now we can build the interface for the API that will actually

从服务器获取数据。

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

这里内容比较紧凑，我们来逐步拆解说明：

- @GET 注解来自 Retrofit，告诉库我们定义的是一个 GET 请求。
- 括号中的路径是我们的 GET 请求应该访问的端点（我们稍后会设置基础 URL）。
- 大括号允许我们在运行时替换路径中的部分内容，从而传递参数。
- 我们定义的函数名为 getDevice，接收我们想要的设备 ID 作为参数。
- @PATH 注解告诉 Retrofit 该参数应替换路径中的 "deviceId" 占位符。
- 该函数返回一个类型为 Device 的 Call 对象。

创建一个包装类：

现在我们将为我们的 API 制作一个小的包装类，以便将 Retrofit 初始化代码很好地封装起来。

```
public class DeviceAPIHelper
{
    public final DeviceAPI api;

    private DeviceAPIHelper ()
    {

        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://example.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();

        api = retrofit.create(DeviceAPI.class);
    }
}
```

该类创建了一个 GSON 实例以解析 JSON 响应，使用我们的基础 url 和 GsonConverter 创建了一个 Retrofit 实例，然后创建了我们的 API 实例。

调用 API：

```
// 获取一个JSON对象
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
            Device device = response.body();
        }
    }
})
```

fetch the data from the server.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

There's a lot going on here in a pretty compact space so let's break it down:

- The @GET annotation comes from Retrofit and tells the library that we're defining a GET request.
- The path in the parentheses is the endpoint that our GET request should hit (we'll set the base url a little later).
- The curly-brackets allow us to replace parts of the path at run time so we can pass arguments.
- The function we're defining is called getDevice and takes the device id we want as an argument.
- The @PATH annotation tells Retrofit that this argument should replace the "deviceId" placeholder in the path.
- The function returns a Call object of type Device.

Creating a wrapper class:

Now we will make a little wrapper class for our API to keep the Retrofit initialization code wrapped up nicely.

```
public class DeviceAPIHelper
{
    public final DeviceAPI api;

    private DeviceAPIHelper ()
    {

        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://example.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();

        api = retrofit.create(DeviceAPI.class);
    }
}
```

This class creates a GSON instance to be able to parse the JSON response, creates a Retrofit instance with our base url and a GsonConverter and then creates an instance of our API.

Calling the API:

```
// Getting a JSON object
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
            Device device = response.body();
        }
    }
})
```

```

@Override
public void onFailure (Call<Device> call, Throwable t)
{
Log.e(TAG, t.getLocalizedMessage());
}

// 获取一个 JSON 数组
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>>() {
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
Log.e(TAG, t.getLocalizedMessage());
    }
});

```

这使用我们的API接口分别创建一个Call<Device>对象和一个Call<List<Device>>对象。
调用enqueue会告诉Retrofit在后台线程执行该调用，并将结果返回到我们这里创建的回调中。

注意：解析原始对象的JSON数组（如String、Integer、Boolean和Double）类似于解析JSON数组。然而，你不需要自己的模型类。例如，可以通过将调用的返回类型设为Call<List<String>>来获取字符串数组。

第96.2节：使用Stetho调试

向你的应用程序添加以下依赖项。

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

在你的Application类的onCreate方法中，调用以下代码。

```
Stetho.initializeWithDefaults(this);
```

创建Retrofit实例时，创建一个自定义的OkHttp实例。

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

然后在 Retrofit 实例中设置这个自定义的 OkHttp 实例。

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

```

@Override
public void onFailure (Call<Device> call, Throwable t)
{
Log.e(TAG, t.getLocalizedMessage());
}

// Getting a JSON array
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>>() {
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
Log.e(TAG, t.getLocalizedMessage());
    }
});

```

This uses our API interface to create a Call<Device> object and to create a Call<List<Device>> respectively.
Calling enqueue tells Retrofit to make that call on a background thread and return the result to the callback that we're creating here.

Note: Parsing a JSON array of primitive objects (like *String*, *Integer*, *Boolean*, and *Double*) is similar to parsing a JSON array. However, you don't need your own model class. You can get the array of *Strings* for example by having the return type of the call as Call<List<String>>.

Section 96.2: Debugging with Stetho

Add the following dependencies to your application.

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

In your Application class' onCreate method, call the following.

```
Stetho.initializeWithDefaults(this);
```

When creating your Retrofit instance, create a custom OkHttp instance.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Then set this custom OkHttp instance in the Retrofit instance.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

现在将手机连接到电脑，启动应用程序，在Chrome浏览器中输入chrome://inspect。Retrofit的网络请求现在应该会显示，供你检查。

第96.3节：为Retrofit2添加日志记录

可以使用拦截器记录Retrofit请求。有几种详细级别可选：NONE、BASIC、HEADERS、BODY。请参见Github项目链接。

- 在build.gradle中添加依赖：

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

- 创建Retrofit时添加日志拦截器：

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
    .addInterceptor(loggingInterceptor)
.build();
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .client(okHttpClient)
.addConverterFactory(GsonConverterFactory.create(gson))
.build();
```

在终端（Android Monitor）中暴露日志在发布版本中应避免，因为这可能导致敏感信息（如认证令牌等）被意外暴露。

为避免运行时日志被暴露，请检查以下条件

```
if(BuildConfig.DEBUG){
    //在此处编写你的拦截器代码
}
```

例如：

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
if(BuildConfig.DEBUG){
    //在此情况下打印日志
    loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
} else{
    loggingInterceptor.setLevel(LoggingInterceptor.Level.NONE);
}

OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
    .addInterceptor(loggingInterceptor)
.build();

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .client(okHttpClient)
.addConverterFactory(GsonConverterFactory.create(gson))
.build();
```

第96.4节：使用GSON的简单POST请求

示例JSON：

Now connect your phone to your computer, launch the app, and type chrome://inspect into your Chrome browser. Retrofit network calls should now show up for you to inspect.

Section 96.3: Add logging to Retrofit2

Retrofit requests can be logged using an interceptor. There are several levels of detail available: NONE, BASIC, HEADERS, BODY. See [Github project here](#).

- Add dependency to build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

- Add logging interceptor when creating Retrofit:

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
    .addInterceptor(loggingInterceptor)
    .build();
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .client(okHttpClient)
.addConverterFactory(GsonConverterFactory.create(gson))
.build();
```

Exposing the logs in the Terminal(Android Monitor) is something that should be avoided in the release version as it may lead to unwanted exposing of critical information such as Auth Tokens etc.

To avoid the logs being exposed in the run time, check the following condition

```
if(BuildConfig.DEBUG){
    //your interceptor code here
}
```

For example:

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
if(BuildConfig.DEBUG){
    //print the logs in this case
    loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
} else{
    loggingInterceptor.setLevel(LoggingInterceptor.Level.NONE);
}

OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
    .addInterceptor(loggingInterceptor)
    .build();

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .client(okHttpClient)
.addConverterFactory(GsonConverterFactory.create(gson))
.build();
```

Section 96.4: A simple POST request with GSON

Sample JSON:

```
{  
    "id": "12345",  
    "type": "android"  
}
```

定义您的请求：

```
public class GetDeviceRequest {  
  
    @SerializedName("deviceId")  
    private String mDeviceId;  
  
    public GetDeviceRequest(String deviceId) {  
        this.mDeviceId = deviceId;  
    }  
  
    public String getDeviceId() {  
        return mDeviceId;  
    }  
}
```

定义您的服务（要访问的端点）：

```
public interface Service {  
  
    @POST("device")  
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);  
}
```

定义网络客户端的单例实例：

```
public class RestClient {  
  
    private static Service REST_CLIENT;  
  
    static {  
        setupRestClient();  
    }  
  
    private static void setupRestClient() {  
  
        // 定义 gson  
        Gson gson = new Gson();  
  
        // 定义我们的客户端  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl("http://example.com/")  
            .addConverterFactory(GsonConverterFactory.create(gson))  
            .build();  
  
        REST_CLIENT = retrofit.create(Service.class);  
    }  
  
    public static Retrofit getRestClient() {  
        return REST_CLIENT;  
    }  
}
```

```
{  
    "id": "12345",  
    "type": "android"  
}
```

Define your request:

```
public class GetDeviceRequest {  
  
    @SerializedName("deviceId")  
    private String mDeviceId;  
  
    public GetDeviceRequest(String deviceId) {  
        this.mDeviceId = deviceId;  
    }  
  
    public String getDeviceId() {  
        return mDeviceId;  
    }  
}
```

Define your service (endpoints to hit):

```
public interface Service {  
  
    @POST("device")  
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);  
}
```

Define your singleton instance of the network client:

```
public class RestClient {  
  
    private static Service REST_CLIENT;  
  
    static {  
        setupRestClient();  
    }  
  
    private static void setupRestClient() {  
  
        // Define gson  
        Gson gson = new Gson();  
  
        // Define our client  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl("http://example.com/")  
            .addConverterFactory(GsonConverterFactory.create(gson))  
            .build();  
  
        REST_CLIENT = retrofit.create(Service.class);  
    }  
  
    public static Retrofit getRestClient() {  
        return REST_CLIENT;  
    }  
}
```

}

定义一个简单的设备模型对象：

```
public class Device {
    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }
}
```

定义控制器以处理设备的请求

```
public class DeviceController {
    // 其他初始化代码...

    public void getDeviceFromAPI() {
        // 定义我们的请求并加入队列
        Call<Device> call = RestClient.getRestClient().getDevice(new GetDeviceRequest("12345"));

        // 继续将请求加入队列
        call.enqueue(new Callback<Device>() {
            @Override
            public void onSuccess(Response<Device> deviceResponse) {
                // 在这里处理您的设备
                if (deviceResponse.isSuccess()) {
                    // 处理成功
                    //delegate.passDeviceObject();
                }
            }

            @Override
            public void onFailure(Throwable t) {
                // 继续在这里处理错误
            }
        });
    }
};
```

第96.5节：使用Retrofit2从服务器下载文件

下载文件的接口声明

```
public interface ApiInterface {
    @GET("movie/now_playing")
    Call<MovieResponse> getNowPlayingMovies(@Query("api_key") String apiKey, @Query("page") int page);
```

}

Define a simple model object for the device:

```
public class Device {
    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }
}
```

Define controller to handle the requests for the device

```
public class DeviceController {
    // Other initialization code here...

    public void getDeviceFromAPI() {
        // Define our request and enqueue
        Call<Device> call = RestClient.getRestClient().getDevice(new GetDeviceRequest("12345"));

        // Go ahead and enqueue the request
        call.enqueue(new Callback<Device>() {
            @Override
            public void onSuccess(Response<Device> deviceResponse) {
                // Take care of your device here
                if (deviceResponse.isSuccess()) {
                    // Handle success
                    //delegate.passDeviceObject();
                }
            }

            @Override
            public void onFailure(Throwable t) {
                // Go ahead and handle the error here
            }
        });
    }
};
```

Section 96.5: Download a file from Server using Retrofit2

Interface declaration for downloading a file

```
public interface ApiInterface {
    @GET("movie/now_playing")
    Call<MovieResponse> getNowPlayingMovies(@Query("api_key") String apiKey, @Query("page") int page);
```

```

// 选项1：相对于你的基础URL的资源
@GET("resource/example.zip")
Call<ResponseBody> downloadFileWithFixedUrl();

// 选项2：使用动态URL
@GET
Call<ResponseBody> downloadFileWithDynamicUrl(@Url String fileUrl);
}

```

选项1用于从具有固定URL的服务器下载文件。选项2用于传递动态值作为完整URL进行请求调用。当下载依赖于用户或时间等参数的文件时，这非常有用。

设置Retrofit以进行API调用

```

public class ServiceGenerator {

    public static final String API_BASE_URL = "http://your.api-base.url/";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
        new Retrofit.Builder()
            .baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass){
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }

}

```

现在，实现用于从服务器下载文件的API

```

private void downloadFile(){
    ApiInterface apiInterface = ServiceGenerator.createService(ApiInterface.class);

    Call<ResponseBody> call = apiInterface.downloadFileWithFixedUrl();

    call.enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
            if (response.isSuccessful()){
                boolean writtenToDisk = writeResponseBodyToDisk(response.body());

                Log.d("文件下载成功?", String.valueOf(writtenToDisk));
            }
        }

        @Override
        public void onFailure(Call<ResponseBody> call, Throwable t) {
        });
}

```

在回调中获取响应后，编写一些标准的IO代码将文件保存到磁盘。代码如下：

```
private boolean writeResponseBodyToDisk(ResponseBody body) {
```

```

// option 1: a resource relative to your base URL
@GET("resource/example.zip")
Call<ResponseBody> downloadFileWithFixedUrl();

// option 2: using a dynamic URL
@GET
Call<ResponseBody> downloadFileWithDynamicUrl(@Url String fileUrl);
}

```

The option 1 is used for downloading a file from Server which is having fixed URL. and option 2 is used to pass a dynamic value as full URL to request call. This can be helpful when downloading files, which are dependent of parameters like user or time.

Setup retrofit for making api calls

```

public class ServiceGenerator {

    public static final String API_BASE_URL = "http://your.api-base.url/";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
        new Retrofit.Builder()
            .baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass){
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }

}

```

Now, make implementation of api for downloading file from server

```

private void downloadFile(){
    ApiInterface apiInterface = ServiceGenerator.createService(ApiInterface.class);

    Call<ResponseBody> call = apiInterface.downloadFileWithFixedUrl();

    call.enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
            if (response.isSuccessful()){
                boolean writtenToDisk = writeResponseBodyToDisk(response.body());

                Log.d("File download was a success? ", String.valueOf(writtenToDisk));
            }
        }

        @Override
        public void onFailure(Call<ResponseBody> call, Throwable t) {
        });
}

```

And after getting response in the callback, code some standard IO for saving file to disk. Here is the code:

```
private boolean writeResponseBodyToDisk(ResponseBody body) {
```

```

try {
    // todo 根据需要更改文件位置/名称
    File futureStudioIconFile = new File(getExternalFilesDir(null) + File.separator +
"Future Studio Icon.png");

    InputStream inputStream = null;
    OutputStream outputStream = null;

    try {
        byte[] fileReader = new byte[4096];

        long fileSize = body.contentLength();
        long fileSizeDownloaded = 0;

        inputStream = body.byteStream();
        outputStream = new FileOutputStream(futureStudioIconFile);

        while (true) {
            int read = inputStream.read(fileReader);

            if (read == -1) {
                break;
            }

            outputStream.write(fileReader, 0, read);

            fileSizeDownloaded += read;

            Log.d("文件下载: " , fileSizeDownloaded + " / " + fileSize);
        }
    }

    outputStream.flush();

    return true;
} catch (IOException e) {
    return false;
} 最后 {
    if (inputStream != null) {
        inputStream.close();
    }

    if (outputStream != null) {
        outputStream.close();
    }
} catch (IOException e) {
    return false;
}
}

```

注意我们已将 `ResponseBody` 指定为返回类型，否则 `Retrofit` 会尝试解析和转换它，这在下载文件时没有意义。

如果你想了解更多关于 `Retrofit` 的内容，可以访问这个链接，非常有用。[1]:

<https://futurestud.io/blog/retrofit-getting-started-and-android-client>

第96.6节：使用 `Retrofit` 作为多部分上传多个文件

一旦你在项目中设置好 `Retrofit` 环境，就可以使用以下示例演示如何使用 `Retrofit` 上传多个文件：

```

try {
    // todo change the file location/name according to your needs
    File futureStudioIconFile = new File(getExternalFilesDir(null) + File.separator +
"Future Studio Icon.png");

    InputStream inputStream = null;
    OutputStream outputStream = null;

    try {
        byte[] fileReader = new byte[4096];

        long fileSize = body.contentLength();
        long fileSizeDownloaded = 0;

        inputStream = body.byteStream();
        outputStream = new FileOutputStream(futureStudioIconFile);

        while (true) {
            int read = inputStream.read(fileReader);

            if (read == -1) {
                break;
            }

            outputStream.write(fileReader, 0, read);

            fileSizeDownloaded += read;

            Log.d("File Download: " , fileSizeDownloaded + " of " + fileSize);
        }
    }

    outputStream.flush();

    return true;
} catch (IOException e) {
    return false;
} finally {
    if (inputStream != null) {
        inputStream.close();
    }

    if (outputStream != null) {
        outputStream.close();
    }
} catch (IOException e) {
    return false;
}
}

```

Note we have specified `ResponseBody` as return type, otherwise `Retrofit` will try to parse and convert it, which doesn't make sense when you are downloading file.

If you want more on `Retrofit` stuffs, got to this link as it is very useful. [1]:

<https://futurestud.io/blog/retrofit-getting-started-and-android-client>

Section 96.6: Upload multiple file using Retrofit as multipart

Once you have setup the `Retrofit` environment in your project, you can use the following example that demonstrates how to upload multiple files using `Retrofit`:

```

private void 多文件上传文件(Uri[] 文件Uri) {
    OkHttpClient okHttpClient = new OkHttpClient();
    OkHttpClient 30秒超时客户端 = okHttpClient.newBuilder()
        .readTimeout(30, TimeUnit.SECONDS)
    .build();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(API_URL_BASE)
    .addConverterFactory(new MultiPartConverter())
        .client(30秒超时客户端)
    .build();

    WebAPIService service = retrofit.create(WebAPIService.class); //这里是创建的调用服务接口

    Map<String, okhttp3.RequestBody> maps = new HashMap<>();

    if (文件Uri!=null && 文件Uri.length>0) {
        for (int i = 0; i < 文件Uri.length; i++) {
            String 文件路径 = getRealPathFromUri(文件Uri[i]);
            File 文件1 = new File(文件路径);

            if (文件路径 != null && 文件路径.length() > 0) {
                if (文件1.exists()) {
                    okhttp3.RequestBody 请求文件 =
                    okhttp3.RequestBody.create(okhttp3.MediaType.parse("multipart/form-data"), 文件1);
                    String 文件名 = "imagePath" + i; //上传文件的键名, 如 : imagePath
                    maps.put(文件名 + "\", 文件名=\"" + 文件1.getName(), 请求文件);
                }
            }
        }

        String descriptionString = " string request";//  

        //hear is the your json request
        Call<String> call = service.postFile(maps, descriptionString);
        call.enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call,
                                  Response<String> response) {
                Log.i(LOG_TAG, "success");
                Log.d("body==>", response.body().toString() + "");
                Log.d("isSuccessful==>", response.isSuccessful() + "");
                Log.d("message==>", response.message() + "");
                Log.d("raw==>", response.raw().toString() + "");
                Log.d("raw().networkResponse()", response.raw().networkResponse().toString() + "");
            }

            @Override
            public void onFailure(Call<String> call, Throwable t) {
                Log.e(LOG_TAG, t.getMessage());
            }
        });
    }

    public String getRealPathFromUri(final Uri uri) { // function for file path from uri,
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
            // ExternalStorageProvider
            if (isExternalStorageDocument(uri)) {
                final String docId = DocumentsContract.getDocumentId(uri);
                final String[] split = docId.split(":");
                final String type = split[0];

```

```

private void mulitpleFileUploadFile(Uri[] fileUri) {
    OkHttpClient okHttpClient = new OkHttpClient();
    OkHttpClient clientWith30sTimeout = okHttpClient.newBuilder()
        .readTimeout(30, TimeUnit.SECONDS)
    .build();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(API_URL_BASE)
    .addConverterFactory(new MultiPartConverter())
        .client(clientWith30sTimeout)
    .build();

    WebAPIService service = retrofit.create(WebAPIService.class); //here is the interface which you
have created for the call service
    Map<String, okhttp3.RequestBody> maps = new HashMap<>();

    if (fileUri!=null && fileUri.length>0) {
        for (int i = 0; i < fileUri.length; i++) {
            String filePath = getRealPathFromUri(fileUri[i]);
            File file1 = new File(filePath);

            if (filePath != null && filePath.length() > 0) {
                if (file1.exists()) {
                    okhttp3.RequestBody requestFile =
                    okhttp3.RequestBody.create(okhttp3.MediaType.parse("multipart/form-data"), file1);
                    String filename = "imagePath" + i; //key for upload file like : imagePath
                    maps.put(filename + "\", filename=\"" + file1.getName(), requestFile);
                }
            }
        }

        String descriptionString = " string request";//  

        //hear is the your json request
        Call<String> call = service.postFile(maps, descriptionString);
        call.enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call,
                                  Response<String> response) {
                Log.i(LOG_TAG, "success");
                Log.d("body==>", response.body().toString() + "");
                Log.d("isSuccessful==>", response.isSuccessful() + "");
                Log.d("message==>", response.message() + "");
                Log.d("raw==>", response.raw().toString() + "");
                Log.d("raw().networkResponse()", response.raw().networkResponse().toString() + "");
            }

            @Override
            public void onFailure(Call<String> call, Throwable t) {
                Log.e(LOG_TAG, t.getMessage());
            }
        });

        public String getRealPathFromUri(final Uri uri) { // function for file path from uri,
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
                // ExternalStorageProvider
                if (isExternalStorageDocument(uri)) {
                    final String docId = DocumentsContract.getDocumentId(uri);
                    final String[] split = docId.split(":");
                    final String type = split[0];

```

```

        if ("primary".equalsIgnoreCase(type)) {
            return Environment.getExternalStorageDirectory() + "/" + split[1];
        }
    } // DownloadsProvider
    else if (isDownloadsDocument(uri)) {

        final String id = DocumentsContract.getDocumentId(uri);
        final Uri contentUri = ContentUris.withAppendedId(
            Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

        return getDataColumn(mContext, contentUri, null, null);
    } // 媒体提供者
    else if (isMediaDocument(uri)) {
        final String docId = DocumentsContract.getDocumentId(uri);
        final String[] split = docId.split(":");
        final String type = split[0];

        Uri contentUri = null;
        if ("image".equals(type)) {
            contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        } else if ("video".equals(type)) {
            contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
        } else if ("audio".equals(type)) {
            contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        }

        final String selection = "_id=?";
        final String[] selectionArgs = new String[]{
            split[1]
        };

        return getDataColumn(mContext, contentUri, selection, selectionArgs);
    } // MediaStore (及通用)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // 返回远程地址
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    } // 文件
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}

```

以下是接口

```

public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}

```

```

        if ("primary".equalsIgnoreCase(type)) {
            return Environment.getExternalStorageDirectory() + "/" + split[1];
        }
    } // DownloadsProvider
    else if (isDownloadsDocument(uri)) {

        final String id = DocumentsContract.getDocumentId(uri);
        final Uri contentUri = ContentUris.withAppendedId(
            Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

        return getDataColumn(mContext, contentUri, null, null);
    } // MediaProvider
    else if (isMediaDocument(uri)) {
        final String docId = DocumentsContract.getDocumentId(uri);
        final String[] split = docId.split(":");
        final String type = split[0];

        Uri contentUri = null;
        if ("image".equals(type)) {
            contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        } else if ("video".equals(type)) {
            contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
        } else if ("audio".equals(type)) {
            contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        }

        final String selection = "_id=?";
        final String[] selectionArgs = new String[]{
            split[1]
        };

        return getDataColumn(mContext, contentUri, selection, selectionArgs);
    } // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    } // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}

```

Following is the interface

```

public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}

```

第96.7节：使用OkHttp拦截器的Retrofit

本示例展示了如何使用OkHttp的请求拦截器。这有多种使用场景，例如：

- 为请求添加通用header。例如，认证请求
- 调试网络应用
- 获取原始response
- 记录网络事务等。
- 设置自定义用户代理

```
Retrofit.Builder builder = new Retrofit.Builder()
    .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
    .addConverterFactory(GsonConverterFactory.create())
    .baseUrl("https://api.github.com/");

if (!TextUtils.isEmpty(githubToken)) {
    // `githubToken`: GitHub的访问令牌
    OkHttpClient client = new OkHttpClient.Builder().addInterceptor(new Interceptor() {
        @Override public Response intercept(Chain chain) throws IOException {
            Request request = chain.request();
            Request newReq = request.newBuilder()
                .addHeader("Authorization", format("token %s", githubToken))
                .build();
            return chain.proceed(newReq);
        }
    }).build();

    builder.client(client);
}

return builder.build().create(GithubApi.class);
```

详情请参见 [OkHttp](#) 主题。

第 96.8 节：头部和主体：一个认证示例

@Header 和 @Body 注解可以放在方法签名中，Retrofit 会根据你的模型自动创建它们。

```
public interface MyService {
    @POST("authentication/user")
    Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
        @Header("Authorization") String basicToken);
}
```

AuthenticationRequest 是我们的模型，一个 POJO，包含服务器所需的信息。对于这个例子，我们的服务器需要客户端密钥和密钥密码。

```
public class AuthenticationRequest {
    String clientKey;
    String clientSecret;
}
```

注意在 @Header("Authorization") 中，我们指定了填充 Authorization 头。其他头部会自动填充，因为 Retrofit 可以根据我们发送和期望返回的对象类型推断它们。

Section 96.7: Retrofit with OkHttp interceptor

This example shows how to use a request interceptor with OkHttp. This has numerous use cases such as:

- Adding universal header to the request. E.g. authenticating a request
- Debugging networked applications
- Retrieving raw response
- Logging network transaction etc.
- Set custom user agent

```
Retrofit.Builder builder = new Retrofit.Builder()
    .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
    .addConverterFactory(GsonConverterFactory.create())
    .baseUrl("https://api.github.com/");

if (!TextUtils.isEmpty(githubToken)) {
    // `githubToken`: Access token for GitHub
    OkHttpClient client = new OkHttpClient.Builder().addInterceptor(new Interceptor() {
        @Override public Response intercept(Chain chain) throws IOException {
            Request request = chain.request();
            Request newReq = request.newBuilder()
                .addHeader("Authorization", format("token %s", githubToken))
                .build();
            return chain.proceed(newReq);
        }
    }).build();

    builder.client(client);
}

return builder.build().create(GithubApi.class);
```

See [OkHttp](#) topic for more details.

Section 96.8: Header and Body: an Authentication Example

The @Header and @Body annotations can be placed into the method signatures and Retrofit will automatically create them based on your models.

```
public interface MyService {
    @POST("authentication/user")
    Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
        @Header("Authorization") String basicToken);
}
```

AuthenticaionRequest 是我们的模型，一个 POJO，包含服务器所需的信息。For this example, our server wants the client key and secret.

```
public class AuthenticationRequest {
    String clientKey;
    String clientSecret;
}
```

Notice that in @Header("Authorization") we are specifying we are populating the Authorization header. The other headers will be populated automatically since Retrofit can infer what they are based on the type of objects we are sending and expecting in return.

我们在某处创建 Retrofit 服务。我们确保使用 HTTPS。

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https:// some example site")
    .client(client)
.build();
MyService myService = retrofit.create(MyService.class)
```

然后我们可以使用我们的服务。

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

第96.9节：通过Multipart上传文件

使用Retrofit2注解声明你的接口：

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ") RequestBody
photo);
}
```

其中RestApiDefaultResponse是包含响应的自定义类。

构建您的API实现并将调用入队：

```
Retrofit retrofit = new Retrofit.Builder()
    .addConverterFactory(GsonConverterFactory.create())
    .baseUrl("http://<yourhost>/")
.client(okHttpClient)
.build();

BackendApiClient apiClient = retrofit.create(BackendApiClient.class);
RequestBody reqBody = RequestBody.create(MediaType.parse("image/jpeg"), photoFile);
Call<RestApiDefaultResponse> call = apiClient.uploadPhoto(reqBody);
call.enqueue(<your callback function>);
```

第96.10节：Retrofit 2 自定义Xml转换器

将依赖项添加到build.gradle文件中。

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ...
}
```

然后创建转换器工厂

We create our Retrofit service somewhere. We make sure to use HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https:// some example site")
    .client(client)
.build();
MyService myService = retrofit.create(MyService.class)
```

Then we can use our service.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Section 96.9: Uploading a file via Multipart

Declare your interface with Retrofit2 annotations:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ") RequestBody
photo);
}
```

Where RestApiDefaultResponse is a custom class containing the response.

Building the implementation of your API and enqueue the call:

```
Retrofit retrofit = new Retrofit.Builder()
    .addConverterFactory(GsonConverterFactory.create())
    .baseUrl("http://<yourhost>/")
    .client(okHttpClient)
    .build();

BackendApiClient apiClient = retrofit.create(BackendApiClient.class);
RequestBody reqBody = RequestBody.create(MediaType.parse("image/jpeg"), photoFile);
Call<RestApiDefaultResponse> call = apiClient.uploadPhoto(reqBody);
call.enqueue(<your callback function>);
```

Section 96.10: Retrofit 2 Custom Xml Converter

Adding dependencies into the build.gradle file.

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ...
}
```

Then create Converter Factory

```

public class XStreamXmlConverterFactory extends Converter.Factory {

    /** 使用默认的 {@link com.thoughtworks.xstream.XStream} 实例进行转换来创建实例。 */
    public static XStreamXmlConverterFactory create() {
        return create(new XStream());
    }

    /** 使用 {@code xStream} 进行转换来创建实例。 */
    public static XStreamXmlConverterFactory create(XStream xStream) {
        return new XStreamXmlConverterFactory(xStream);
    }

    private final XStream xStream;

    private XStreamXmlConverterFactory(XStream xStream) {
        if (xStream == null) throw new NullPointerException("xStream == null");
        this.xStream = xStream;
    }

    @Override
    public Converter<ResponseBody, ?> responseBodyConverter(Type type, Annotation[] annotations,
Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        Class<?> cls = (Class<?>) type;

        return new XStreamXmlResponseBodyConverter<>(cls, xStream);
    }

    @Override
    public Converter<?, RequestBody> requestBodyConverter(Type type,
        Annotation[] parameterAnnotations, Annotation[] methodAnnotations, Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        return new XStreamXmlRequestBodyConverter<>(xStream);
    }
}

```

创建一个类来处理请求体。

```

final class XStreamXmlResponseBodyConverter <T> implements Converter<ResponseBody, T> {

    private final Class<T> cls;
    private final XStream xStream;

    XStreamXmlResponseBodyConverter(Class<T> cls, XStream xStream) {
        this.cls = cls;
        this.xStream = xStream;
    }

    @Override
    public T convert(ResponseBody value) throws IOException {
        try {

```

```

public class XStreamXmlConverterFactory extends Converter.Factory {

    /** Create an instance using a default {@link com.thoughtworks.xstream.XStream} instance for
conversion. */
    public static XStreamXmlConverterFactory create() {
        return create(new XStream());
    }

    /** Create an instance using {@code xStream} for conversion. */
    public static XStreamXmlConverterFactory create(XStream xStream) {
        return new XStreamXmlConverterFactory(xStream);
    }

    private final XStream xStream;

    private XStreamXmlConverterFactory(XStream xStream) {
        if (xStream == null) throw new NullPointerException("xStream == null");
        this.xStream = xStream;
    }

    @Override
    public Converter<ResponseBody, ?> responseBodyConverter(Type type, Annotation[] annotations,
Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        Class<?> cls = (Class<?>) type;

        return new XStreamXmlResponseBodyConverter<>(cls, xStream);
    }

    @Override
    public Converter<?, RequestBody> requestBodyConverter(Type type,
        Annotation[] parameterAnnotations, Annotation[] methodAnnotations, Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        return new XStreamXmlRequestBodyConverter<>(xStream);
    }
}

```

create a class to handle the body request.

```

final class XStreamXmlResponseBodyConverter <T> implements Converter<ResponseBody, T> {

    private final Class<T> cls;
    private final XStream xStream;

    XStreamXmlResponseBodyConverter(Class<T> cls, XStream xStream) {
        this.cls = cls;
        this.xStream = xStream;
    }

    @Override
    public T convert(ResponseBody value) throws IOException {
        try {

```

```

    this.xStream.processAnnotations(cls);
    Object object = this.xStream.fromXML(value.byteStream());
    return (T) object;

}finally {
value.close();
}
}
}

```

创建一个类来处理主体响应。

```

final class XStreamXmlRequestBodyConverter<T> implements Converter<T, RequestBody> {

    private static final MediaType MEDIA_TYPE = MediaType.parse("application/xml; charset=UTF-8");
    private static final String CHARSET = "UTF-8";

    private final XStream xStream;

    XStreamXmlRequestBodyConverter(XStream xStream) {
        this.xStream = xStream;
    }

    @Override
    public RequestBody convert(T value) throws IOException {

        Buffer buffer = new Buffer();

        try {
            OutputStreamWriter osw = new OutputStreamWriter(buffer.outputStream(), CHARSET);
            xStream.toXML(value, osw);
            osw.flush();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return RequestBody.create(MEDIA_TYPE, buffer.readByteString());
    }
}

```

所以，到这里我们可以发送和接收任何XML，我们只需要为实体创建XStream注解。

然后创建一个Retrofit实例：

```

XStream xs = new XStream(new DomDriver());
xs.autodetectAnnotations(true);

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .addConverterFactory(XStreamXmlConverterFactory.create(xs))
    .client(client)
    .build();

```

第96.11节：使用Retrofit 2从URL读取XML表单

我们将使用Retrofit 2和SimpleXmlConverter从URL获取XML数据并解析为Java类。

向Gradle脚本添加依赖：

```

    this.xStream.processAnnotations(cls);
    Object object = this.xStream.fromXML(value.byteStream());
    return (T) object;

}finally {
    value.close();
}
}
}

```

create a class to handle the body response.

```

final class XStreamXmlRequestBodyConverter<T> implements Converter<T, RequestBody> {

    private static final MediaType MEDIA_TYPE = MediaType.parse("application/xml; charset=UTF-8");
    private static final String CHARSET = "UTF-8";

    private final XStream xStream;

    XStreamXmlRequestBodyConverter(XStream xStream) {
        this.xStream = xStream;
    }

    @Override
    public RequestBody convert(T value) throws IOException {

        Buffer buffer = new Buffer();

        try {
            OutputStreamWriter osw = new OutputStreamWriter(buffer.outputStream(), CHARSET);
            xStream.toXML(value, osw);
            osw.flush();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return RequestBody.create(MEDIA_TYPE, buffer.readByteString());
    }
}

```

So, this point we can send and receive any XML , We just need create XStream Annotations for the entities.

Then create a Retrofit instance:

```

XStream xs = new XStream(new DomDriver());
xs.autodetectAnnotations(true);

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .addConverterFactory(XStreamXmlConverterFactory.create(xs))
    .client(client)
    .build();

```

Section 96.11: Reading XML from URL with Retrofit 2

We will use retrofit 2 and SimpleXmlConverter to get xml data from url and parse to Java class.

Add dependency to Gradle script:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'  
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

创建接口

同时创建XML类包装器，在我们的例子中是Rss类

```
public interface ApiDataInterface{  
  
    // 网站上XML链接的路径  
  
    @GET ("data/read.xml")  
    Call<Rss> getData();  
}
```

Xml读取函数

```
private void readXmlFeed() {  
    try {  
  
        // 基础网址 - 网站的URL  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl("http://www.google.com/").  
            .client(new OkHttpClient())  
            .addConverterFactory(SimpleXmlConverterFactory.create())  
            .build();  
  
        ApiDataInterface apiService = retrofit.create(ApiDataInterface.class);  
  
        Call<Rss> call = apiService.getData();  
        call.enqueue(new Callback<Rss>() {  
  
            @Override  
            public void onResponse(Call<Rss> call, Response<Rss> response) {  
  
                Log.e("Response success", response.message());  
  
            }  
  
            @Override  
            public void onFailure(Call<Rss> call, Throwable t) {  
                Log.e("Response fail", t.getMessage());  
            }  
        });  
  
    } catch (Exception e) {  
        Log.e("Exception", e.getMessage());  
    }  
}
```

这是一个带有SimpleXML注解的Java类示例

更多关于注解 [SimpleXmlDocumentation](#)

```
@Root (name = "rss")
```

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'  
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Create interface

Also create xml class wrapper in our case Rss class

```
public interface ApiDataInterface{  
  
    // path to xml link on web site  
  
    @GET ("data/read.xml")  
    Call<Rss> getData();  
}
```

Xml read function

```
private void readXmlFeed() {  
    try {  
  
        // base url - url of web site  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl("http://www.google.com/").  
            .client(new OkHttpClient())  
            .addConverterFactory(SimpleXmlConverterFactory.create())  
            .build();  
  
        ApiDataInterface apiService = retrofit.create(ApiDataInterface.class);  
  
        Call<Rss> call = apiService.getData();  
        call.enqueue(new Callback<Rss>() {  
  
            @Override  
            public void onResponse(Call<Rss> call, Response<Rss> response) {  
  
                Log.e("Response success", response.message());  
  
            }  
  
            @Override  
            public void onFailure(Call<Rss> call, Throwable t) {  
                Log.e("Response fail", t.getMessage());  
            }  
        });  
  
    } catch (Exception e) {  
        Log.e("Exception", e.getMessage());  
    }  
}
```

This is example of Java class with SimpleXML annotations

More about annotations [SimpleXmlDocumentation](#)

```
@Root (name = "rss")
```

```
public class Rss
{
    public Rss() {
    }

    public Rss(String title, String description, String link, List<Item> item, String language) {
        this.title = title;
        this.description = description;
        this.link = link;
        this.item = item;
        this.language = language;
    }

    @Element(name = "title")
    private String title;

    @Element(name = "description")
    private String description;

    @Element(name = "link")
    private String link;

    @ElementList(entry="item", inline=true)
    private List<Item> item;

    @Element(name = "language")
    private String language;
}
```

```
public class Rss
{
    public Rss() {
    }

    public Rss(String title, String description, String link, List<Item> item, String language) {
        this.title = title;
        this.description = description;
        this.link = link;
        this.item = item;
        this.language = language;
    }

    @Element(name = "title")
    private String title;

    @Element(name = "description")
    private String description;

    @Element(name = "link")
    private String link;

    @ElementList(entry="item", inline=true)
    private List<Item> item;

    @Element(name = "language")
    private String language;
}
```

第97章：ButterKnife

ButterKnife 是一个视图绑定工具，使用注解为我们生成样板代码。该工具由 Square 的 Jake Wharton 开发，主要用于节省编写重复代码行的时间，例如使用 `findViewById(R.id.view)` 来处理视图，使我们的代码看起来更加简洁。

明确一点，Butterknife 不是依赖注入库。Butterknife 在编译时注入代码。它与 Android Annotations 所做的工作非常相似。

第97.1节：在项目中配置 ButterKnife

配置你的项目级 build.gradle 以包含 android-apt 插件：

```
buildscript {  
    repositories {  
        mavenCentral()  
    }  
  
dependencies {  
    classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'  
}  
}
```

然后，在你的模块级 build.gradle 中应用 android-apt 插件，并添加 ButterKnife 依赖：

```
apply plugin: 'android-apt'  
  
android {  
    ...  
}  
  
dependencies {  
    compile 'com.jakewharton:butterknife:8.5.1'  
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'  
}
```

注意：如果您使用的是版本2.2.0或更新版本的新Jack编译器，则不需要 android-apt 插件，在声明编译器依赖时可以用 annotationProcessor 替代 apt。

为了使用ButterKnife注解，您不要忘记在Activity的onCreate()方法或Fragment的onCreateView()方法中绑定它们：

```
class ExampleActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // 绑定注解  
        ButterKnife.bind(this);  
        // ...  
    }  
  
    // 或者  
    class ExampleFragment extends Fragment {  
        ...  
    }  
}
```

Chapter 97: ButterKnife

Butterknife is a view binding tool that uses annotations to generate boilerplate code for us. This tool is developed by Jake Wharton at Square and is essentially used to save typing repetitive lines of code like `findViewById(R.id.view)` when dealing with views thus making our code look a lot cleaner.

To be clear, Butterknife is **not a dependency injection library**. Butterknife injects code at compile time. It is very similar to the work done by Android Annotations.

Section 97.1: Configuring ButterKnife in your project

Configure your project-level build.gradle to include the android-apt plugin:

```
buildscript {  
    repositories {  
        mavenCentral()  
    }  
  
dependencies {  
    classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'  
}
```

Then, apply the android-apt plugin in your module-level build.gradle and add the ButterKnife dependencies:

```
apply plugin: 'android-apt'  
  
android {  
    ...  
}  
  
dependencies {  
    compile 'com.jakewharton:butterknife:8.5.1'  
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'  
}
```

Note: If you are using the new Jack compiler with version 2.2.0 or newer you do not need the android-apt plugin and can instead replace apt with annotationProcessor when declaring the compiler dependency.

In order to use ButterKnife annotations you shouldn't forget about binding them in `onCreate()` of your Activities or `onCreateView()` of your Fragments:

```
class ExampleActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // Binding annotations  
        ButterKnife.bind(this);  
        // ...  
    }  
  
    // Or  
    class ExampleFragment extends Fragment {  
        ...  
    }  
}
```

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(getContentView(), container, false);
    // 绑定注解
    ButterKnife.bind(this, view);
    ...
    return view;
}

```

开发版本的快照可在Sonatype的快照仓库中获取。

以下是您在库项目中使用ButterKnife需要采取的额外步骤

要在库项目中使用ButterKnife，请将插件添加到项目级的build.gradle中：

```

buildscript {
dependencies {
classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
}
}

```

...然后通过在库级别的build.gradle顶部添加以下行来应用到您的模块：

```

apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'

```

现在确保在所有ButterKnife注解中使用R2而不是R。

```

class ExampleActivity extends Activity {

    // 将xml资源绑定到它们的视图
    @BindView(R2.id.user) EditText 用户名;
    @BindView(R2.id.pass) EditText 密码;

    // 绑定drawable、strings、dimens、colors资源
    @BindString(R.string.choose) String 选择;
    @BindDrawable(R.drawable.send) Drawable 发送;
    @BindColor(R.color.cyan) int 青色;
    @BindDimen(R.dimen.margin) float 通用边距;

    // 监听器
    @OnClick(R.id.submit)
    public void 提交(View 视图) {
        // TODO 提交数据到服务器...
    }

    // 在onCreate中使用butterknife绑定
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        // TODO 继续
    }
}

```

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(getContentView(), container, false);
    // Binding annotations
    ButterKnife.bind(this, view);
    ...
    return view;
}

```

Snapshots of the development version are available in [Sonatype's snapshots repository](#).

Below are the additional steps you'd have to take to use ButterKnife in a library project

To use ButterKnife in a library project, add the plugin to your project-level build.gradle:

```

buildscript {
dependencies {
classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
}
}

```

...and then apply to your module by adding these lines on the top of your library-level build.gradle:

```

apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'

```

Now make sure you use R2 instead of R inside all ButterKnife annotations.

```

class ExampleActivity extends Activity {

    // Bind xml resource to their View
    @BindView(R2.id.user) EditText username;
    @BindView(R2.id.pass) EditText password;

    // Binding resources from drawable,strings,dimens,colors
    @BindString(R.string.choose) String choose;
    @BindDrawable(R.drawable.send) Drawable send;
    @BindColor(R.color.cyan) int cyan;
    @BindDimen(R.dimen.margin) float generalMargin;

    // Listeners
    @OnClick(R.id.submit)
    public void submit(View view) {
        // TODO submit data to server...
    }

    // bind with butterknife in onCreate
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        // TODO continue
    }
}

```

第97.2节：ButterKnife中解绑视图

Fragment的视图生命周期与Activity不同。在onCreateView中绑定Fragment时，应在onDestroyView中将视图设置为null。调用bind时，Butter Knife会返回一个Unbinder实例帮你完成此操作。在合适的生命周期回调中调用它的unbind方法。

示例：

```
public class MyFragment extends Fragment {
    @BindView(R.id.textView) TextView textView;
    @BindView(R.id.button) Button button;
    private Unbinder unbinder;

    @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.my_fragment, container, false);
        unbinder = ButterKnife.bind(this, view);
        // TODO 使用字段...
        return view;
    }

    @Override public void onDestroyView() {
        super.onDestroyView();
        unbinder.unbind();
    }
}
```

注意：在onDestroyView()中调用unbind()并非必须，但推荐这样做，因为如果你的应用有较大的返回栈，这样可以节省相当多的内存。

Section 97.2: Unbinding views in ButterKnife

Fragments have a different view lifecycle than activities. When binding a fragment in onCreateView, set the views to null in onDestroyView. Butter Knife returns an Unbinder instance when you call bind to do this for you. Call its unbind method in the appropriate lifecycle callback.

An example:

```
public class MyFragment extends Fragment {
    @BindView(R.id.textView) TextView textView;
    @BindView(R.id.button) Button button;
    private Unbinder unbinder;

    @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.my_fragment, container, false);
        unbinder = ButterKnife.bind(this, view);
        // TODO Use fields...
        return view;
    }

    @Override public void onDestroyView() {
        super.onDestroyView();
        unbinder.unbind();
    }
}
```

Note: Calling unbind() in onDestroyView() is not required, but recommended as it saves quite a bit of memory if your app has a large backstack.

第 97.3 节：使用 ButterKnife 绑定监听器

点击监听器：

```
@OnClick(R.id.login)
public void login(View view) {
    // 额外逻辑
}
```

监听器方法的所有参数都是可选的：

```
@OnClick(R.id.login)
public void login() {
    // 额外逻辑
}
```

特定类型会被自动转换：

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
    button.setText("Hello!");
}
```

Section 97.3: Binding Listeners using ButterKnife

onClick Listener:

```
@OnClick(R.id.login)
public void login(View view) {
    // Additional logic
}
```

All arguments to the listener method are optional:

```
@OnClick(R.id.login)
public void login() {
    // Additional logic
}
```

Specific type will be automatically casted:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
    button.setText("Hello!");
}
```

单个绑定中多个ID用于通用事件处理：

```
@OnClick({ R.id.door1, R.id.door2, R.id.door3 })
public void pickDoor(DoorView door) {
    if (door.hasPrizeBehind()) {
        Toast.makeText(this, "你赢了!", LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "再试一次", LENGTH_SHORT).show();
    }
}
```

自定义视图可以通过不指定ID来绑定它们自己的监听器：

```
public class CustomButton extends Button {
    @OnClick
    public void onClick() {
        // TODO
    }
}
```

第97.4节：Android Studio ButterKnife 插件

Android ButterKnife Zelezny

用于从活动/片段/适配器中选定的布局XML生成ButterKnife注入的插件。

注意：请确保你对 `your_xml_layout(R.layout.your_xml_layout)` 进行了正确的右键点击，否则生成菜单中将不包含ButterKnife注入选项。

Multiple IDs in a single binding for common event handling:

```
@OnClick({ R.id.door1, R.id.door2, R.id.door3 })
public void pickDoor(DoorView door) {
    if (door.hasPrizeBehind()) {
        Toast.makeText(this, "You win!", LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "Try again", LENGTH_SHORT).show();
    }
}
```

Custom Views can bind to their own listeners by not specifying an ID:

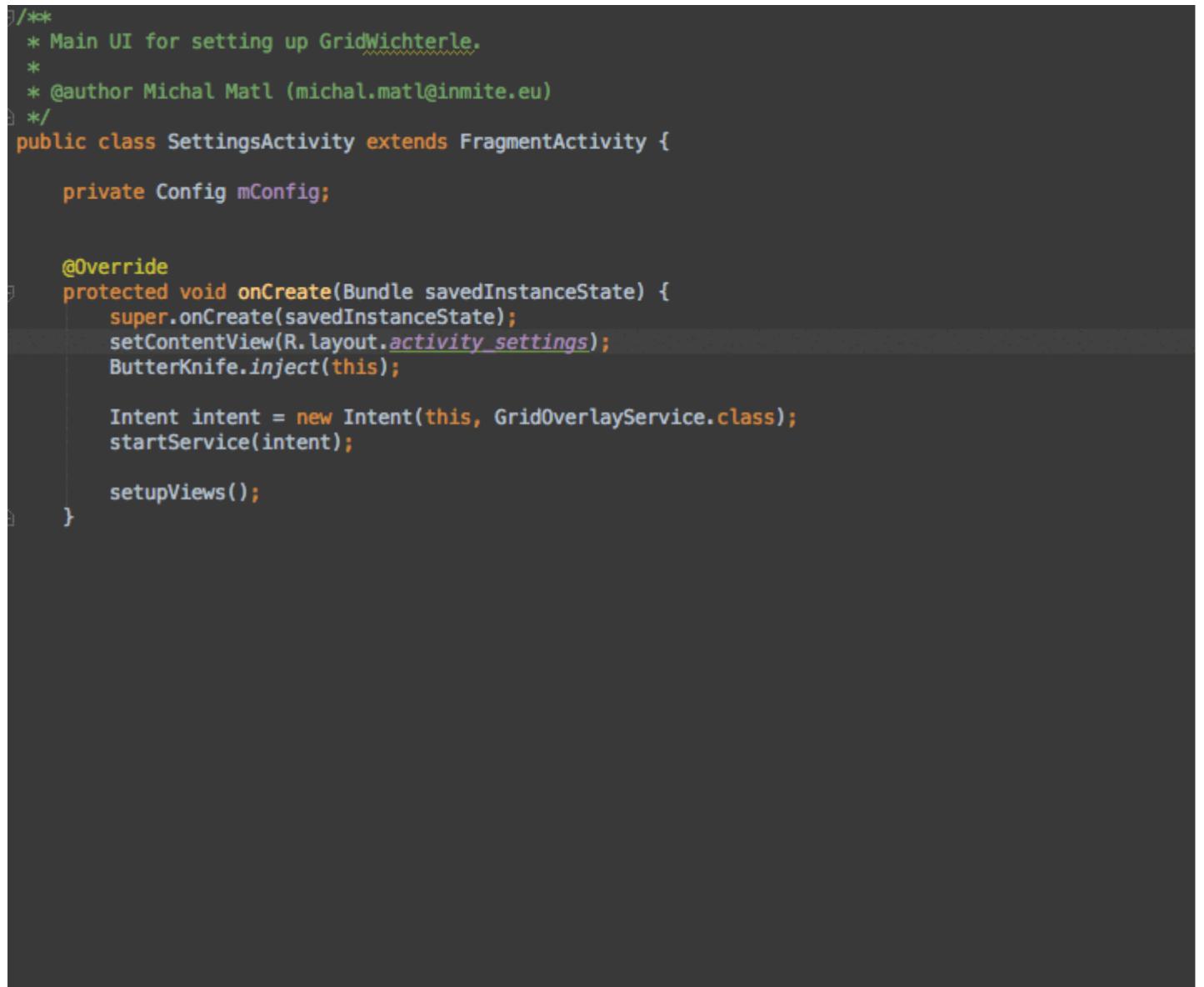
```
public class CustomButton extends Button {
    @OnClick
    public void onClick() {
        // TODO
    }
}
```

Section 97.4: Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

Plugin for generating ButterKnife injections from selected layout XMLs in activities/fragments/adapters.

Note : Make sure that you make the right click for `your_xml_layout(R.layout.your_xml_layout)` else the `Generate menu` will not contain Butterknife injector option.



链接：[Jetbrains 插件 Android ButterKnife Zelezny](#)

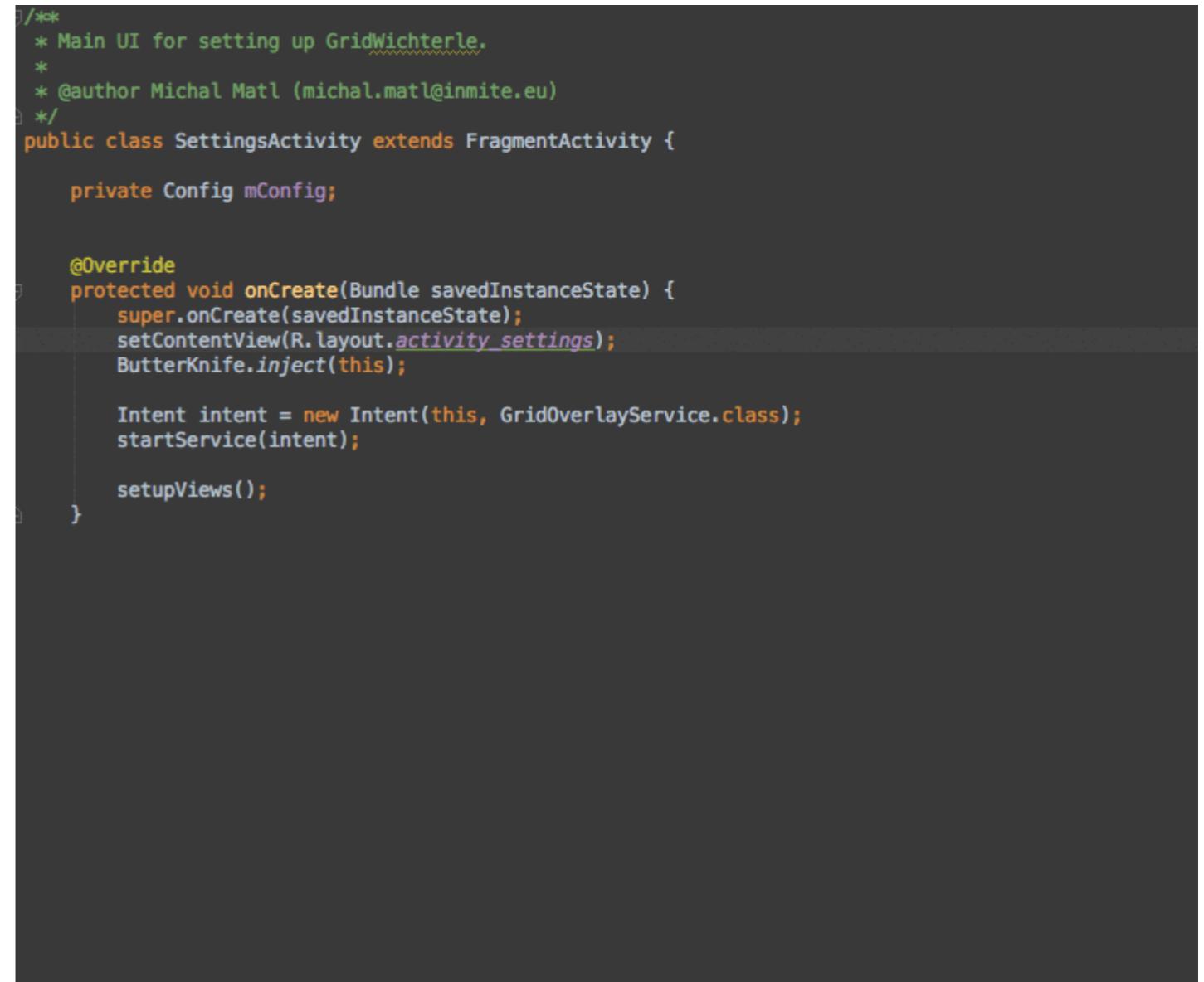
第97.5节：使用ButterKnife绑定视图

我们可以使用@BindView注解字段并指定视图ID，让ButterKnife查找并自动转换布局中对应的视图。

绑定视图

在Activity中绑定视图

```
类 ExampleActivity extends Activity {  
    @BindView(R.id.title) TextView title;  
    @BindView(R.id.subtitle) TextView subtitle;  
    @BindView(R.id.footer) TextView footer;  
  
    @Override public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.simple_activity);  
        ButterKnife.bind(this);  
        // TODO 使用字段...  
    }  
}
```



Link：[Jetbrains Plugin Android ButterKnife Zelezny](#)

Section 97.5: Binding Views using ButterKnife

we can annotate fields with @BindView and a view ID for Butter Knife to find and automatically cast the corresponding view in our layout.

Binding Views

Binding Views in Activity

```
class ExampleActivity extends Activity {  
    @BindView(R.id.title) TextView title;  
    @BindView(R.id.subtitle) TextView subtitle;  
    @BindView(R.id.footer) TextView footer;  
  
    @Override public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.simple_activity);  
        ButterKnife.bind(this);  
        // TODO Use fields...  
    }  
}
```

在碎片中绑定视图

```
public class FancyFragment extends Fragment {
    @BindView(R.id.button1) Button button1;
    @BindView(R.id.button2) Button button2;
    private Unbinder unbinder;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fancy_fragment, container, false);
        unbinder = ButterKnife.bind(this, view);
        // TODO 使用字段...
        return view;
    }

    // 在碎片或非活动绑定中，当视图即将被销毁时，我们需要解绑绑定
    @Override
    public void onDestroy() {
        super.onDestroy();
        unbinder.unbind();
    }
}
```

在对话框中绑定视图

我们可以使用ButterKnife.findById在视图、活动或对话框中查找视图。它使用泛型推断返回类型并自动进行类型转换。

```
View view = LayoutInflater.from(context).inflate(R.layout.thing, null);
TextView firstName = ButterKnife.findById(view, R.id.first_name);
TextView lastName = ButterKnife.findById(view, R.id.last_name);
ImageView photo = ButterKnife.findById(view, R.id.photo);
```

在 ViewHolder 中绑定视图

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
        ButterKnife.bind(this, view);
    }
}
```

绑定资源

除了用于绑定视图之外，还可以使用 ButterKnife 绑定资源，例如在 strings.xml、drawables.xml、colors.xml、dimens.xml 等文件中定义的资源。

```
public class ExampleActivity extends Activity {

    @BindString(R.string.title) String title;
    @BindDrawable(R.drawable.graphic) Drawable graphic;
    @BindColor(R.color.red) int red; // int or ColorStateList field
    @BindDimen(R.dimen.spacer) float spacer; // int (for pixel size) or float (for exact value)
    field

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ...
    }
}
```

Binding Views in Fragments

```
public class FancyFragment extends Fragment {
    @BindView(R.id.button1) Button button1;
    @BindView(R.id.button2) Button button2;
    private Unbinder unbinder;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fancy_fragment, container, false);
        unbinder = ButterKnife.bind(this, view);
        // TODO Use fields...
        return view;
    }

    // in fragments or non activity bindings we need to unbind the binding when view is about to be destroyed
    @Override
    public void onDestroy() {
        super.onDestroy();
        unbinder.unbind();
    }
}
```

Binding Views in Dialogs

We can use ButterKnife.findById to find views on a View, Activity, or Dialog. It uses generics to infer the return type and automatically performs the cast.

```
View view = LayoutInflater.from(context).inflate(R.layout.thing, null);
TextView firstName = ButterKnife.findById(view, R.id.first_name);
TextView lastName = ButterKnife.findById(view, R.id.last_name);
ImageView photo = ButterKnife.findById(view, R.id.photo);
```

Binding Views in ViewHolder

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
        ButterKnife.bind(this, view);
    }
}
```

Binding Resources

Apart from being useful for binding views, one could also use ButterKnife to bind resources such as those defined within strings.xml, drawables.xml, colors.xml, dimens.xml, etc.

```
public class ExampleActivity extends Activity {

    @BindString(R.string.title) String title;
    @BindDrawable(R.drawable.graphic) Drawable graphic;
    @BindColor(R.color.red) int red; // int or ColorStateList field
    @BindDimen(R.dimen.spacer) float spacer; // int (for pixel size) or float (for exact value)
    field

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ...
    }
}
```

```
ButterKnife.bind(this);  
}  
}
```

绑定视图列表

您可以将多个视图分组到一个列表或数组中。当我们需要对多个视图同时执行某个操作时，这非常有用。

```
@BindViews({ R.id.first_name, R.id.middle_name, R.id.last_name })
```

```
List<EditText> nameViews;
```

//apply 方法允许您一次对列表中的所有视图执行操作。

```
ButterKnife.apply(nameViews, DISABLE);  
ButterKnife.apply(nameViews, ENABLED, false);
```

//我们可以使用 Action 和 Setter 接口来指定简单的行为。

```
static final ButterKnife.Action<View> DISABLE = new ButterKnife.Action<View>() {  
    @Override public void apply(View view, int index) {  
        view.setEnabled(false);  
    }  
};  
static final ButterKnife.Setter<View, Boolean> ENABLED = new ButterKnife.Setter<View, Boolean>() {  
    @Override public void set(View view, Boolean value, int index) {  
        view.setEnabled(value);  
    }  
};
```

可选绑定

默认情况下，@Bind 和监听器绑定都是必需的。如果找不到目标视图，会抛出异常。但如果不确定视图是否存在，则可以在字段上添加@Nullable注解，或在方法上添加@Optional注解，以抑制此行为并创建可选绑定。

```
@Nullable  
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;
```

```
@Optional  
@OnClick(R.id.maybe_missing)  
void onMaybeMissingClicked() {  
    // TODO ...  
}
```

```
ButterKnife.bind(this);  
}
```

```
}
```

Binding View Lists

You can group multiple views into a List or array. This is very helpful when we need to perform one action on multiple views at once.

```
@BindViews({ R.id.first_name, R.id.middle_name, R.id.last_name })  
List<EditText> nameViews;
```

//The apply method allows you to act on all the views in a list at once.

```
ButterKnife.apply(nameViews, DISABLE);  
ButterKnife.apply(nameViews, ENABLED, false);
```

//We can use Action and Setter interfaces allow specifying simple behavior.

```
static final ButterKnife.Action<View> DISABLE = new ButterKnife.Action<View>() {  
    @Override public void apply(View view, int index) {  
        view.setEnabled(false);  
    }  
};  
static final ButterKnife.Setter<View, Boolean> ENABLED = new ButterKnife.Setter<View, Boolean>() {  
    @Override public void set(View view, Boolean value, int index) {  
        view.setEnabled(value);  
    }  
};
```

Optional Bindings

By default, both @Bind and listener bindings are required. An exception is thrown if the target view cannot be found. But if we are not sure if a view will be there or not then we can add a @Nullable annotation to fields or the @Optional annotation to methods to suppress this behavior and create an optional binding.

```
@Nullable  
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;
```

```
@Optional  
@OnClick(R.id.maybe_missing)  
void onMaybeMissingClicked() {  
    // TODO ...  
}
```

第98章：Volley

Volley是谷歌推出的一个Android HTTP库，旨在简化网络请求。默认情况下，所有Volley网络请求都是异步进行的，所有操作都在后台线程处理，并通过回调在前台返回结果。由于通过网络获取数据是任何应用中最常见的任务之一，Volley库的出现极大地简化了Android应用开发。

第98.1节：使用Volley进行HTTP请求

在应用级 `build.gradle` 中添加 `gradle` 依赖

```
compile 'com.android.volley:volley:1.0.0'
```

另外，在应用的清单文件中添加 `android.permission.INTERNET` 权限。

在你的 Application 中创建 Volley RequestQueue 实例单例

```
public class InitApplication extends Application {

    private RequestQueue queue;
    private static InitApplication sInstance;

    private static final String TAG = InitApplication.class.getSimpleName();

    @Override
    public void onCreate() {
        super.onCreate();

        sInstance = this;

        Stetho.initializeWithDefaults(this);
    }

    public static synchronized InitApplication getInstance() {
        return sInstance;
    }

    public <T> void addToQueue(Request<T> req, String tag) {
        req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
        getQueue().add(req);
    }

    public <T> void addToQueue(Request<T> req) {
        req.setTag(TAG);
        getQueue().add(req);
    }

    public void 取消挂起请求(Object 标签) {
        if (队列 != null) {
            队列.取消全部(标签);
        }
    }

    public 请求队列 获取队列() {
        if (队列 == null) {
            队列 = Volley.新建请求队列(获取应用上下文());
            返回 队列;
        }
    }
}
```

Chapter 98: Volley

Volley is an Android HTTP library that was introduced by Google to make networking calls much simpler. By default all the Volley network calls are made asynchronously, handling everything in a background thread and returning the results in the foreground with use of callbacks. As fetching data over a network is one of the most common tasks that is performed in any app, the Volley library was made to ease Android app development.

Section 98.1: Using Volley for HTTP requests

Add the gradle dependency in app-level build.gradle

```
compile 'com.android.volley:volley:1.0.0'
```

Also, add the `android.permission.INTERNET` permission to your app's manifest.

**Create Volley RequestQueue instance singleton in your Application **

```
public class InitApplication extends Application {

    private RequestQueue queue;
    private static InitApplication sInstance;

    private static final String TAG = InitApplication.class.getSimpleName();

    @Override
    public void onCreate() {
        super.onCreate();

        sInstance = this;

        Stetho.initializeWithDefaults(this);
    }

    public static synchronized InitApplication getInstance() {
        return sInstance;
    }

    public <T> void addToQueue(Request<T> req, String tag) {
        req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
        getQueue().add(req);
    }

    public <T> void addToQueue(Request<T> req) {
        req.setTag(TAG);
        getQueue().add(req);
    }

    public void cancelPendingRequests(Object tag) {
        if (queue != null) {
            queue.cancelAll(tag);
        }
    }

    public RequestQueue getQueue() {
        if (queue == null) {
            queue = Volley.newRequestQueue(getApplicationContext());
        }
        return queue;
    }
}
```

```
    }
    返回队列;
}
}
```

现在，你可以使用 `getInstance()` 方法获取 volley 实例，并使用以下方式将新请求添加到队列中
`InitApplication.getInstance().addRequest(request);`

一个从服务器请求 `JSONObject` 的简单示例是

```
JsonObjectRequest myRequest = new JsonObjectRequest(Method.GET,
        url, null,
        new Response.Listener<JSONObject>() {

            @Override
            public void onResponse(JSONObject response) {
                Log.d(TAG, response.toString());
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d(TAG, "Error: " + error.getMessage());
            }
        });

myRequest.setRetryPolicy(new DefaultRetryPolicy(
        MY_SOCKET_TIMEOUT_MS,
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
```

要处理 Volley 超时，您需要使用 `RetryPolicy`。重试策略用于在请求因网络故障或其他情况无法完成时进行处理。

Volley 提供了一种简便的方法来为您的请求实现 `RetryPolicy`。默认情况下，Volley 将所有请求的套接字和连接超时设置为 5 秒。`RetryPolicy` 是一个接口，您需要在其中实现当超时发生时如何重试特定请求的逻辑。

构造函数接受以下三个参数：

- `initialTimeoutMs` - 指定每次重试尝试的套接字超时时间，单位为毫秒。
- `maxNumRetries` - 重试尝试的次数。
- `backoffMultiplier` - 用于确定每次重试时套接字的指数时间设置的乘数。

第98.2节：使用GET方法的基本StringRequest

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// 实例化RequestQueue。
RequestQueue queue = Volley.newRequestQueue(this);
String url = "http://www.google.com";

// 从提供的URL请求字符串响应。
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
    @Override
```

```
    }
    return queue;
}
}
```

Now, you can use the volley instance using the `getInstance()` method and add a new request in the queue using `InitApplication.getInstance().addRequest(request);`

A simple example to request `JSONObject` from server is

```
JsonObjectRequest myRequest = new JsonObjectRequest(Method.GET,
        url, null,
        new Response.Listener<JSONObject>() {

            @Override
            public void onResponse(JSONObject response) {
                Log.d(TAG, response.toString());
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d(TAG, "Error: " + error.getMessage());
            }
        });

myRequest.setRetryPolicy(new DefaultRetryPolicy(
        MY_SOCKET_TIMEOUT_MS,
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
```

To handle Volley timeouts you need to use a `RetryPolicy`. A retry policy is used in case a request cannot be completed due to network failure or some other cases.

Volley provides an easy way to implement your `RetryPolicy` for your requests. By default, Volley sets all socket and connection timeouts to 5 seconds for all requests. `RetryPolicy` is an interface where you need to implement your logic of how you want to retry a particular request when a timeout occurs.

The constructor takes the following three parameters:

- `initialTimeoutMs` - Specifies the socket timeout in milliseconds for every retry attempt.
- `maxNumRetries` - The number of times retry is attempted.
- `backoffMultiplier` - A multiplier which is used to determine exponential time set to socket for every retry attempt.

Section 98.2: Basic StringRequest using GET method

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url = "http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
    @Override
```

```

public void onResponse(String response) {
    // 显示响应字符串的前500个字符。
    mTextView.setText("响应是：" + response.substring(0,500));
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
    mTextView.setText("操作失败！");
}
});
// 将请求添加到请求队列中。
queue.add(stringRequest);

```

第98.3节：向NetworkImageView添加自定义设计时属性

Volley的NetworkImageView为标准的ImageView添加了几个额外的属性。然而，这些属性只能在代码中设置。以下是一个示例，展示如何创建一个扩展类，该类将从你的XML布局文件中获取属性并应用到NetworkImageView实例上。

在你的~/res/xml目录下，添加一个名为attrx.xml的文件：

```

<resources>
<declare-styleable name="MoreNetworkImageView">
    <attr name="defaultImageResId" format="reference"/>
    <attr name="errorImageResId" format="reference"/>
</declare-styleable>
</resources>

```

向你的项目中添加一个新的类文件：

```

package my.namespace;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.NonNull;
import android.util.AttributeSet;

import com.android.volley.toolbox.NetworkImageView;

public class MoreNetworkImageView extends NetworkImageView {
    public MoreNetworkImageView(@NonNull final Context context) {
        super(context);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs)
    {
        super(context, attrs, 0);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs,
final int defStyle) {
        super(context, attrs, defStyle);

        final TypedArray attributes = context.obtainStyledAttributes(attrs,
R.styleable.MoreNetworkImageView, defStyle, 0);

        // 从XML加载defaultImageResId
        int defaultImageResId =

```

```

public void onResponse(String response) {
    // Display the first 500 characters of the response string.
    mTextView.setText("Response is：" + response.substring(0,500));
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
    mTextView.setText("That didn't work!");
}
});
// Add the request to the RequestQueue.
queue.add(stringRequest);

```

Section 98.3: Adding custom design time attributes to NetworkImageView

There are several additional attributes that the Volley NetworkImageView adds to the standard ImageView. However, these attributes can only be set in code. The following is an example of how to make an extension class that will pick up the attributes from your XML layout file and apply them to the NetworkImageView instance for you.

In your ~/res/xml directory, add a file named attrx.xml:

```

<resources>
<declare-styleable name="MoreNetworkImageView">
    <attr name="defaultImageResId" format="reference"/>
    <attr name="errorImageResId" format="reference"/>
</declare-styleable>
</resources>

```

Add a new class file to your project:

```

package my.namespace;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.NonNull;
import android.util.AttributeSet;

import com.android.volley.toolbox.NetworkImageView;

public class MoreNetworkImageView extends NetworkImageView {
    public MoreNetworkImageView(@NonNull final Context context) {
        super(context);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs)
    {
        super(context, attrs, 0);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs,
final int defStyle) {
        super(context, attrs, defStyle);

        final TypedArray attributes = context.obtainStyledAttributes(attrs,
R.styleable.MoreNetworkImageView, defStyle, 0);

        // load defaultImageResId from XML
        int defaultImageResId =

```

```

        attributes.getResourceId(R.styleable.MoreNetworkImageView_defaultImageResId, 0);
        if (defaultImageResId > 0) {
            setDefaultImageResId(defaultImageResId);
        }

        // From XML load errorImageResId
        int errorImageResId =
        attributes.getResourceId(R.styleable.MoreNetworkImageView_errorImageResId, 0);
        if (errorImageResId > 0) {
            setErrorImageResId(errorImageResId);
        }
    }
}

```

一个展示自定义属性使用的示例布局文件：

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent">

    <my.namespace.MoreNetworkImageView
        android:layout_width="64dp"
        android:layout_height="64dp"
        app:errorImageResId="@drawable/error_img"
        app:defaultImageResId="@drawable/default_img"
        tools:defaultImageResId="@drawable/editor_only_default_img"/>
    <!--
注意：“tools:”前缀在Android Studio 2.1及更早版本中至少对自定义属性无效，  

因此在此示例中，defaultImageResId在编辑器中将显示为“default_img”，而不是“editor_only_default_img”  

即使它应该像标准Android属性那样被正确支持为仅编辑器覆盖。  

仅作为编辑器覆盖正确支持的标准Android属性。
-->
</my.namespace.MoreNetworkImageView>
</android.support.v7.widget.CardView>

```

第 98.4 节：向请求添加自定义头部 [例如用于基本认证]

如果你需要为你的 Volley 请求添加自定义头部，初始化之后无法再添加，因为头部信息保存在一个私有变量中。

相反，你需要重写Request类的getHeaders()方法，示例如下：

```

new JsonObjectRequest(REQUEST_METHOD, REQUEST_URL, REQUEST_BODY, RESP_LISTENER, ERR_LISTENER) {
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new HashMap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
};

```

```

        attributes.getResourceId(R.styleable.MoreNetworkImageView_defaultImageResId, 0);
        if (defaultImageResId > 0) {
            setDefaultImageResId(defaultImageResId);
        }

        // Load errorImageResId from XML
        int errorImageResId =
        attributes.getResourceId(R.styleable.MoreNetworkImageView_errorImageResId, 0);
        if (errorImageResId > 0) {
            setErrorImageResId(errorImageResId);
        }
    }
}

```

An example layout file showing the use of the custom attributes:

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent">

    <my.namespace.MoreNetworkImageView
        android:layout_width="64dp"
        android:layout_height="64dp"
        app:errorImageResId="@drawable/error_img"
        app:defaultImageResId="@drawable/default_img"
        tools:defaultImageResId="@drawable/editor_only_default_img"/>
    <!--
Note: The "tools:" prefix does NOT work for custom attributes in Android Studio 2.1 and
older at least, so in this example the defaultImageResId would show "default_img" in the
editor, not the "editor_only_default_img" drawable even though it should if it was
supported as an editor-only override correctly like standard Android properties.
-->
</my.namespace.MoreNetworkImageView>
</android.support.v7.widget.CardView>

```

Section 98.4: Adding custom headers to your requests [e.g. for basic auth]

If you need to add custom headers to your volley requests, you can't do this after initialisation, as the headers are saved in a private variable.

Instead, you need to override the getHeaders() method of Request.class as such:

```

new JsonObjectRequest(REQUEST_METHOD, REQUEST_URL, REQUEST_BODY, RESP_LISTENER, ERR_LISTENER) {
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new HashMap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
};

```

参数说明：

- REQUEST_METHOD - 取值为Request.Method.* 常量之一。
- REQUEST_URL - 发送请求的完整 URL。
- REQUEST_BODY - 一个包含要发送的POST正文的JSONObject（或null）。
- RESP_LISTENER - 一个Response.Listener<?>对象，其onResponse(T data)方法在成功完成时被调用。
- ERR_LISTENER - 一个Response.ErrorListener对象，其onErrorResponse(VolleyError e)方法在请求失败时被调用。

如果你想构建自定义请求，也可以在其中添加请求头：

```
public class MyCustomRequest extends Request {  
    ...  
    @Override  
    public Map<String, String> getHeaders() throws AuthFailureError {  
        HashMap<String, String> customHeaders = new HashMap<>();  
  
        customHeaders.put("KEY_0", "VALUE_0");  
        ...  
        customHeaders.put("KEY_N", "VALUE_N");  
  
        return customHeaders;  
    }  
    ...  
}
```

第98.5节：通过POST方法使用StringRequest进行远程服务器认证

为了这个示例，假设我们有一个服务器来处理我们将从Android应用发出的POST请求：

```
// 用户输入数据。  
String email = "my@email.com";  
String password = "123";  
  
// 我们用于处理POST请求的服务器URL。  
String URL = "http://my.server.com/login.php";  
  
// 当我们使用 Volley 创建一个 StringRequest (或 JSONRequest) 来发送数据时，  
// 我们指定请求方法为 POST，  
// 以及接收我们数据的 URL。  
StringRequest stringRequest =  
    new StringRequest(Request.Method.POST, URL,  
    new Response.Listener<String>() {  
        @Override  
        public void onResponse(String response) {  
            // 此时，Volley 已经将数据发送到你的 URL  
            // 并且收到了响应。我假设  
            // 服务器返回的是一个"OK"字符串。  
            if (response.equals("OK")) {  
                // 执行登录相关操作。  
            } else {  
                // 服务器没有返回"OK"响应。  
                // 根据你在服务器上处理错误的方式，  
                // 你可以决定这里采取什么操作。  
            }  
        }  
    }
```

Explanation of the parameters:

- REQUEST_METHOD - Either of the Request.Method.* constants.
- REQUEST_URL - The full URL to send your request to.
- REQUEST_BODY - A JSONObject containing the POST-Body to be sent (or null).
- RESP_LISTENER - A Response.Listener<?> object, whose onResponse(T data) method is called upon successful completion.
- ERR_LISTENER - A Response.ErrorListener object, whose onErrorResponse(VolleyError e) method is called upon a unsuccessful request.

If you want to build a custom request, you can add the headers in it as well:

```
public class MyCustomRequest extends Request {  
    ...  
    @Override  
    public Map<String, String> getHeaders() throws AuthFailureError {  
        HashMap<String, String> customHeaders = new HashMap<>();  
  
        customHeaders.put("KEY_0", "VALUE_0");  
        ...  
        customHeaders.put("KEY_N", "VALUE_N");  
  
        return customHeaders;  
    }  
    ...  
}
```

Section 98.5: Remote server authentication using StringRequest through POST method

For the sake of this example, let us assume that we have a server for handling the POST requests that we will be making from our Android app:

```
// User input data.  
String email = "my@email.com";  
String password = "123";  
  
// Our server URL for handling POST requests.  
String URL = "http://my.server.com/login.php";  
  
// When we create a StringRequest (or a JSONRequest) for sending  
// data with Volley, we specify the Request Method as POST, and  
// the URL that will be receiving our data.  
StringRequest stringRequest =  
    new StringRequest(Request.Method.POST, URL,  
    new Response.Listener<String>() {  
        @Override  
        public void onResponse(String response) {  
            // At this point, Volley has sent the data to your URL  
            // and has a response back from it. I'm going to assume  
            // that the server sends an "OK" string.  
            if (response.equals("OK")) {  
                // Do login stuff.  
            } else {  
                // So the server didn't return an "OK" response.  
                // Depending on what you did to handle errors on your  
                // server, you can decide what action to take here.  
            }  
        }  
    }
```

```

    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // 这是发生与 Volley 相关错误时的处理。
        // 如果发生这种情况，具体怎么处理取决于你，
        // 但通常不建议向用户过于明确地说明
        // 这里发生了什么。
    }
} {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        // 这里是告诉 Volley 应该在
        // 我们的 POST 请求中发送什么。对于这个例子，我们想要发送
        // 邮箱和密码。
        // 我们需要为数据准备键名，这样服务器才能知道
        // 什么是什么。
        String key_email = "email";
        String key_password = "password";

        Map<String, String> map = new HashMap<String, String>();
        // map.put(key, value);
        map.put(key_email, email);
        map.put(key_password, password);
        return map;
    }
};

// 这是一个策略，我们需要指定它来告诉 Volley 在超时后该做什么，重试多少次等
。
stringRequest.setRetryPolicy(new RetryPolicy() {
    @Override
    public int getCurrentTimeout() {
        // 这里设置超时时间。
        // 数值单位是毫秒，5000 通常足够，
        // 但你可以根据需要调高或调低这个数值。
        return 50000;
    }
    @Override
    public int getCurrentRetryCount() {
        // 最大尝试次数。
        // 同样，这个数值可以根据你的需求设定。
        return 50000;
    }
    @Override
    public void retry(VolleyError error) throws VolleyError {
        // 这里你可以检查重试次数是否达到最大值，
        // 如果达到，可以发送一个 VolleyError
        // 消息或类似处理。为了示例，我将
        // 显示一个 Toast 提示。
        Toast.makeText(getContext(), error.toString(), Toast.LENGTH_LONG).show();
    }
});

// 最后，我们创建一个 Volley 队列。这个例子中我用的是
// getContext()，因为我在使用 Fragment。但 context 也可以是
// "this"、"getContext()" 等。
RequestQueue requestQueue = Volley.newRequestQueue(getContext());
requestQueue.add(stringRequest);

```

```

    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // This is when errors related to Volley happen.
        // It's up to you what to do if that should happen, but
        // it's usually not a good idea to be too clear as to
        // what happened here to your users.
    }
} {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        // Here is where we tell Volley what it should send in
        // our POST request. For this example, we want to send
        // both the email and the password.

        // We will need key ids for our data, so our server can know
        // what is what.
        String key_email = "email";
        String key_password = "password";

        Map<String, String> map = new HashMap<String, String>();
        // map.put(key, value);
        map.put(key_email, email);
        map.put(key_password, password);
        return map;
    }
};

// This is a policy that we need to specify to tell Volley, what
// to do if it gets a timeout, how many times to retry, etc.
stringRequest.setRetryPolicy(new RetryPolicy() {
    @Override
    public int getCurrentTimeout() {
        // Here goes the timeout.
        // The number is in milliseconds, 5000 is usually enough,
        // but you can up or low that number to fit your needs.
        return 50000;
    }
    @Override
    public int getCurrentRetryCount() {
        // The maximum number of attempts.
        // Again, the number can be anything you need.
        return 50000;
    }
    @Override
    public void retry(VolleyError error) throws VolleyError {
        // Here you could check if the retry count has gotten
        // to the maximum number, and if so, send a VolleyError
        // message or similar. For the sake of the example, I'll
        // show a Toast.
        Toast.makeText(getContext(), error.toString(), Toast.LENGTH_LONG).show();
    }
});

// And finally, we create a Volley Queue. For this example, I'm using
// getContext()，because I was working with a Fragment. But context could
// be "this", "getContext()", etc.
RequestQueue requestQueue = Volley.newRequestQueue(getContext());
requestQueue.add(stringRequest);

```

```

} else {
    // 例如, 如果用户输入的邮箱当前不在您的远程数据库中, 您可以在这里通知用户。
    Toast.makeText(getApplicationContext(), "错误的邮箱", Toast.LENGTH_LONG).show();
}

```

第98.6节：取消请求

```

// 假设在上方某处已经初始化了Request和RequestQueue

public static final String TAG = "SomeTag";

// 在请求上设置标签。
request.setTag(TAG);

// 将请求添加到请求队列中。
mRequestQueue.add(request);

// 取消这个特定的请求
request.cancel();

// ... 然后, 在某个未来的生命周期事件中, 例如在 onStop() 中
// 取消 RequestQueue 中所有带有指定标签的请求
mRequestQueue.cancelAll(TAG);

```

第 98.7 节：请求 JSON

```

final TextView mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
ImageView mImageView;
String url = "http://ip.jsontest.com/";

final JsonObjectRequest jsObjRequest = new JsonObjectRequest
    (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            mTxtDisplay.setText("Response: " + response.toString());
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // ...
        }
});

requestQueue.add(jsObjRequest);

```

第98.8节：将JSONArray用作请求体

Volley中默认集成的请求不允许在POST请求中传递JSONArray作为请求体。相反，你只能传递一个JSON对象作为参数。

但是，不是将JSON对象作为参数传递给请求构造函数，而是需要重写Request类的getBody()方法。你还应该将null作为第三个参数传递：

```

JSONArray requestBody = new JSONArray();

new JsonObjectRequest(Request.Method.POST, REQUEST_URL, null, RESP_LISTENER, ERR_LISTENER) {
    @Override

```

```

} else {
    // If, for example, the user inputs an email that is not currently
    // on your remote DB, here's where we can inform the user.
    Toast.makeText(getApplicationContext(), "Wrong email", Toast.LENGTH_LONG).show();
}

```

Section 98.6: Cancel a request

```

// assume a Request and RequestQueue have already been initialized somewhere above

public static final String TAG = "SomeTag";

// Set the tag on the request.
request.setTag(TAG);

// Add the request to the RequestQueue.
mRequestQueue.add(request);

// To cancel this specific request
request.cancel();

// ... then, in some future life cycle event, for example in onStop()
// To cancel all requests with the specified tag in RequestQueue
mRequestQueue.cancelAll(TAG);

```

Section 98.7: Request JSON

```

final TextView mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
ImageView mImageView;
String url = "http://ip.jsontest.com/";

final JsonObjectRequest jsObjRequest = new JsonObjectRequest
    (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            mTxtDisplay.setText("Response: " + response.toString());
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // ...
        }
});

requestQueue.add(jsObjRequest);

```

Section 98.8: Use JSONArray as request body

The default requests integrated in volley don't allow to pass a JSONArray as request body in a POST request. Instead, you can only pass a JSON object as a parameter.

However, instead of passing a JSON object as a parameter to the request constructor, you need to override the getBody() method of the `Request.class`. You should pass `null` as third parameter as well:

```

JSONArray requestBody = new JSONArray();

new JsonObjectRequest(Request.Method.POST, REQUEST_URL, null, RESP_LISTENER, ERR_LISTENER) {
    @Override

```

```

public byte[] getBody() {
    try {
        return requestBody.toString().getBytes(PROTOCOL_CHARSET);
    } catch (UnsupportedEncodingException uee) {
        // 错误处理
        return null;
    }
}

```

参数说明：

- REQUEST_URL - 发送请求的完整 URL。
- RESP_LISTENER - 一个Response.Listener<?>对象，其onResponse(T data)方法在成功完成时被调用。
- ERR_LISTENER - 一个Response.ErrorListener对象，其onErrorResponse(VolleyError e)方法会在请求失败时被调用。

第98.9节：Volley中带有json请求的服务器布尔变量响应

你可以自定义以下类

```

private final String PROTOCOL_CONTENT_TYPE = String.format("application/json; charset=%s",
PROTOCOL_CHARSET);

public BooleanRequest(int method, String url, String requestBody, Response.Listener<Boolean>
listener, Response.ErrorListener errorListener) {
    super(method, url, errorListener);
    this.mListener = listener;
    this.mErrorListener = errorListener;
    this.mRequestBody = requestBody;
}

@Override
protected Response<Boolean> parseNetworkResponse(NetworkResponse response) {
    Boolean parsed;
    try {
parsed = Boolean.valueOf(new String(response.data,
HttpHeaderParser.parseCharset(response.headers)));
    } catch (UnsupportedEncodingException e) {
        parsed = Boolean.valueOf(new String(response.data));
    }
    return Response.success(parsed, HttpHeaderParser.parseCacheHeaders(response));
}

@Override
protected VolleyError parseNetworkError(VolleyError volleyError) {
    return super.parseNetworkError(volleyError);
}

@Override
protected void deliverResponse(Boolean response) {
    mListener.onResponse(response);
}

@Override
public void deliverError(VolleyError error) {
    mErrorListener.onErrorResponse(error);
}

```

```

public byte[] getBody() {
    try {
        return requestBody.toString().getBytes(PROTOCOL_CHARSET);
    } catch (UnsupportedEncodingException uee) {
        // error handling
        return null;
    }
}

```

Explanation of the parameters:

- REQUEST_URL - The full URL to send your request to.
- RESP_LISTENER - A Response.Listener<?> object, whose onResponse(T data) method is called upon successful completion.
- ERR_LISTENER - A Response.ErrorListener object, whose onErrorResponse(VolleyError e) method is called upon an unsuccessful request.

Section 98.9: Boolean variable response from server with json request in volley

you can custom class below one

```

private final String PROTOCOL_CONTENT_TYPE = String.format("application/json; charset=%s",
PROTOCOL_CHARSET);

public BooleanRequest(int method, String url, String requestBody, Response.Listener<Boolean>
listener, Response.ErrorListener errorListener) {
    super(method, url, errorListener);
    this.mListener = listener;
    this.mErrorListener = errorListener;
    this.mRequestBody = requestBody;
}

@Override
protected Response<Boolean> parseNetworkResponse(NetworkResponse response) {
    Boolean parsed;
    try {
parsed = Boolean.valueOf(new String(response.data,
HttpHeaderParser.parseCharset(response.headers)));
    } catch (UnsupportedEncodingException e) {
        parsed = Boolean.valueOf(new String(response.data));
    }
    return Response.success(parsed, HttpHeaderParser.parseCacheHeaders(response));
}

@Override
protected VolleyError parseNetworkError(VolleyError volleyError) {
    return super.parseNetworkError(volleyError);
}

@Override
protected void deliverResponse(Boolean response) {
    mListener.onResponse(response);
}

@Override
public void deliverError(VolleyError error) {
    mErrorListener.onErrorResponse(error);
}

```

```

}

@Override
public String getBodyContentType() {
    return PROTOCOL_CONTENT_TYPE;
}

@Override
public byte[] getBody() throws AuthFailureError {
    try {
        return mRequestBody == null ? null : mRequestBody.getBytes(PROTOCOL_CHARSET);
    } catch (UnsupportedEncodingException uee) {
        VolleyLog.wtf("尝试使用 %s 获取 %s 的字节时不支持的编码",
                     mRequestBody, PROTOCOL_CHARSET);
        return null;
    }
}

```

在你的活动中使用这个

```

try {
    JSONObject jsonBody;
    jsonBody = new JSONObject();
    jsonBody.put("Title", "Android 演示");
    jsonBody.put("Author", "BNK");
    jsonBody.put("Date", "2015/08/28");
    String requestBody = jsonBody.toString();
    BooleanRequest booleanRequest = new BooleanRequest(0, url, requestBody, new
    Response.Listener<Boolean>() {
        @Override
        public void onResponse(Boolean response) {
            Toast.makeText(mContext, String.valueOf(response), Toast.LENGTH_SHORT).show();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(mContext, error.toString(), Toast.LENGTH_SHORT).show();
        }
    });
    // 将请求添加到请求队列中。
    queue.add(booleanRequest);
} catch (JSONException e) {
    e.printStackTrace();
}

```

第98.10节：处理Volley错误的辅助类

```

public class VolleyErrorHelper {
    /**
     * 返回应显示给用户的适当消息
     * 针对指定的错误对象。
     *
     * @param error
     * @param context
     * @return
     */

    public static String getMessage (Object error , Context context){
        if(error instanceof TimeoutError){

```

```

}

@Override
public String getBodyContentType() {
    return PROTOCOL_CONTENT_TYPE;
}

@Override
public byte[] getBody() throws AuthFailureError {
    try {
        return mRequestBody == null ? null : mRequestBody.getBytes(PROTOCOL_CHARSET);
    } catch (UnsupportedEncodingException uee) {
        VolleyLog.wtf("Unsupported Encoding while trying to get the bytes of %s using %s",
                     mRequestBody, PROTOCOL_CHARSET);
        return null;
    }
}

```

use this with your activity

```

try {
    JSONObject jsonBody;
    jsonBody = new JSONObject();
    jsonBody.put("Title", "Android Demo");
    jsonBody.put("Author", "BNK");
    jsonBody.put("Date", "2015/08/28");
    String requestBody = jsonBody.toString();
    BooleanRequest booleanRequest = new BooleanRequest(0, url, requestBody, new
    Response.Listener<Boolean>() {
        @Override
        public void onResponse(Boolean response) {
            Toast.makeText(mContext, String.valueOf(response), Toast.LENGTH_SHORT).show();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(mContext, error.toString(), Toast.LENGTH_SHORT).show();
        }
    });
    // Add the request to the RequestQueue.
    queue.add(booleanRequest);
} catch (JSONException e) {
    e.printStackTrace();
}

```

Section 98.10: Helper Class for Handling Volley Errors

```

public class VolleyErrorHelper {
    /**
     * Returns appropriate message which is to be displayed to the user
     * against the specified error object.
     *
     * @param error
     * @param context
     * @return
     */

    public static String getMessage (Object error , Context context){
        if(error instanceof TimeoutError){

```

```

        return context.getResources().getString(R.string.timeout);
    }else if (isServerProblem(error)){
        return handleServerError(error ,context);
    }else if(isNetworkProblem(error)){
        return context.getResources().getString(R.string.nointernet);
    }
    return context.getString(R.string.generic_error);
}

private static String handleServerError(Object error, Context context) {

VolleyError er = (VolleyError)error;
    NetworkResponse response = er.networkResponse;
    if(response != null){
        switch (response.statusCode){

            case 404:
            case 422:
            case 401:
                try {
                    // 服务器可能返回类似这样的错误 { "error": "Some error occurred"
                }
                // 使用 "Gson" 解析结果
HashMap<String, String> result = new Gson().fromJson(new
String(response.data),
                new TypeToken<Map<String, String>>() {
                    .getType());
                if (result != null && result.containsKey("error")) {
                    return result.get("error");
                }
            } catch (Exception e) {
e.printStackTrace();
            }
            // 无效请求
            return ((VolleyError) error).getMessage();

        default:
            return context.getResources().getString(R.string.timeout);
        }
    }
    return context.getString(R.string.generic_error);
}

private static boolean isServerProblem(Object error) {
    return (error instanceof ServerError || error instanceof AuthFailureError);
}

private static boolean isNetworkProblem (Object error){
    return (error instanceof NetworkError || error instanceof NoConnectionError);
}

```

```

        return context.getResources().getString(R.string.timeout);
    }else if (isServerProblem(error)){
        return handleServerError(error ,context);
    }else if(isNetworkProblem(error)){
        return context.getResources().getString(R.string.nointernet);
    }
    return context.getString(R.string.generic_error);
}

private static String handleServerError(Object error, Context context) {

VolleyError er = (VolleyError)error;
    NetworkResponse response = er.networkResponse;
    if(response != null){
        switch (response.statusCode){

            case 404:
            case 422:
            case 401:
                try {
                    // server might return error like this { "error": "Some error occurred"
                }
                // Use "Gson" to parse the result
HashMap<String, String> result = new Gson().fromJson(new
String(response.data),
                new TypeToken<Map<String, String>>() {
                    .getType());
                if (result != null && result.containsKey("error")) {
                    return result.get("error");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            // invalid request
            return ((VolleyError) error).getMessage();

        default:
            return context.getResources().getString(R.string.timeout);
        }
    }
    return context.getString(R.string.generic_error);
}

private static boolean isServerProblem(Object error) {
    return (error instanceof ServerError || error instanceof AuthFailureError);
}

private static boolean isNetworkProblem (Object error){
    return (error instanceof NetworkError || error instanceof NoConnectionError);
}

```

第99章：日期和时间选择器

第99.1节：日期选择对话框

这是一个对话框，提示用户使用DatePicker选择日期。该对话框需要上下文、初始年份、月份和日期，以便显示带有起始日期的对话框。当用户选择日期时，会通过回调进行处理 DatePickerDialog.OnDateSetListener。

```
public void showDatePicker(Context context, int initialYear, int initialMonth, int initialDay) {  
    DatePickerDialog datePickerDialog = new DatePickerDialog(context,  
        new DatePickerDialog.OnDateSetListener() {  
            @Override  
            public void onDateSet(DatePicker datepicker, int year, int month, int day) {  
                //此条件对于所有安卓版本正常工作是必要的  
                if(view.isShown()){  
                    //您现在拥有选定的年、月和日  
                }  
            }  
        }, initialYear, initialMonth, initialDay);  
  
    //调用 show() 方法即可显示对话框  
datePickerDialog.show();  
}
```

请注意，月份是从0开始的整数，0代表一月，11代表十二月

第99.2节：Material 日期选择器

在build.gradle文件的依赖部分添加以下依赖。（这是一个非官方的日期选择器库）

```
compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

现在我们需要在按钮点击事件中打开DatePicker。

所以在xml文件中创建一个按钮，如下所示。

```
<Button  
    android:id="@+id/dialog_bt_date"  
    android:layout_below="@+id/resetButton"  
    android:layout_width="wrap_content"  
    android:layout_height="40dp"  
    android:textColor="#FF000000"  
    android:gravity="center"  
    android:text="日期"/>
```

然后在MainActivity中这样使用。

```
public class MainActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener{  
  
    Button button;  
    日历 calendar ;  
    日期选择对话框 datePickerDialog ;  
    整数 年, 月, 日 ;
```

Chapter 99: Date and Time Pickers

Section 99.1: Date Picker Dialog

It is a dialog which prompts user to select date using DatePicker. The dialog requires context, initial year, month and day to show the dialog with starting date. When the user selects the date it callbacks via DatePickerDialog.OnDateSetListener.

```
public void showDatePicker(Context context, int initialYear, int initialMonth, int initialDay) {  
    DatePickerDialog datePickerDialog = new DatePickerDialog(context,  
        new DatePickerDialog.OnDateSetListener() {  
            @Override  
            public void onDateSet(DatePicker datepicker, int year, int month, int day) {  
                //this condition is necessary to work properly on all android versions  
                if(view.isShown()){  
                    //You now have the selected year, month and day  
                }  
            }  
        }, initialYear, initialMonth, initialDay);  
  
    //Call show() to simply show the dialog  
datePickerDialog.show();  
}
```

Please note that month is a int starting from 0 for January to 11 for December

Section 99.2: Material DatePicker

add below dependencies to build.gradle file in dependency section. (this is an unOfficial library for date picker)

```
compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Now we have to open DatePicker on Button click event.

So create one Button on xml file like below.

```
<Button  
    android:id="@+id/dialog_bt_date"  
    android:layout_below="@+id/resetButton"  
    android:layout_width="wrap_content"  
    android:layout_height="40dp"  
    android:textColor="#FF000000"  
    android:gravity="center"  
    android:text="DATE"/>
```

and in MainActivity use this way.

```
public class MainActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener{  
  
    Button button;  
    Calendar calendar ;  
    DatePickerDialog datePickerDialog ;  
    int Year, Month, Day ;
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
设置内容视图(R.布局.activity_main);

    calendar = 日历.获取实例();

    年 = calendar.获取(日历.年) ;
    月 = calendar.获取(日历.月);
    日 = calendar.获取(日历.日_当月);

    按钮 dialog_bt_date = (按钮)查找视图通过ID(R.id.dialog_bt_date);
dialog_bt_date.设置点击监听器(新建 视图.点击监听器() {
    @重写
    public void onClick(View view) {

datePickerDialog = 日期选择对话框.新实例(主活动.此, 年, 月,
日);

datePickerDialog.设置暗色主题(假);

    datePickerDialog.首次显示年份选择器(假);

    datePickerDialog.设置强调颜色(颜色.解析颜色("#0072BA"));

    datePickerDialog.设置标题("从日期选择对话框选择日期");

    datePickerDialog.显示(获取碎片管理器(), "DatePickerDialog");
}
}

@Override
公共 无返回值 日期设置回调(日期选择对话框 视图, 整数 年, 整数 月, 整数 日) {

字符串 日期 = "选定日期 : " + 日 + "-" + 月 + "-" + 年;

吐司.制作文本(主活动.此, 日期, 吐司.长时间).显示();
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
getMenuInflater().inflate(R.menu.abc_main_menu, menu);
    return true;
}
}

```

输出：

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    calendar = Calendar.getInstance();

    Year = calendar.get(Calendar.YEAR) ;
    Month = calendar.get(Calendar.MONTH);
    Day = calendar.get(Calendar.DAY_OF_MONTH);

    Button dialog_bt_date = (Button)findViewById(R.id.dialog_bt_date);
dialog_bt_date.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        datePickerDialog = DatePickerDialog.newInstance(MainActivity.this, Year, Month,
Day);

        datePickerDialog.setThemeDark(false);

        datePickerDialog.showYearPickerFirst(false);

        datePickerDialog.setAccentColor(Color.parseColor("#0072BA"));

        datePickerDialog.setTitle("Select Date From DatePickerDialog");

        datePickerDialog.show(getFragmentManager(), "DatePickerDialog");
    }
});

@Override
public void onDateSet(DatePickerDialog view, int Year, int Month, int Day) {

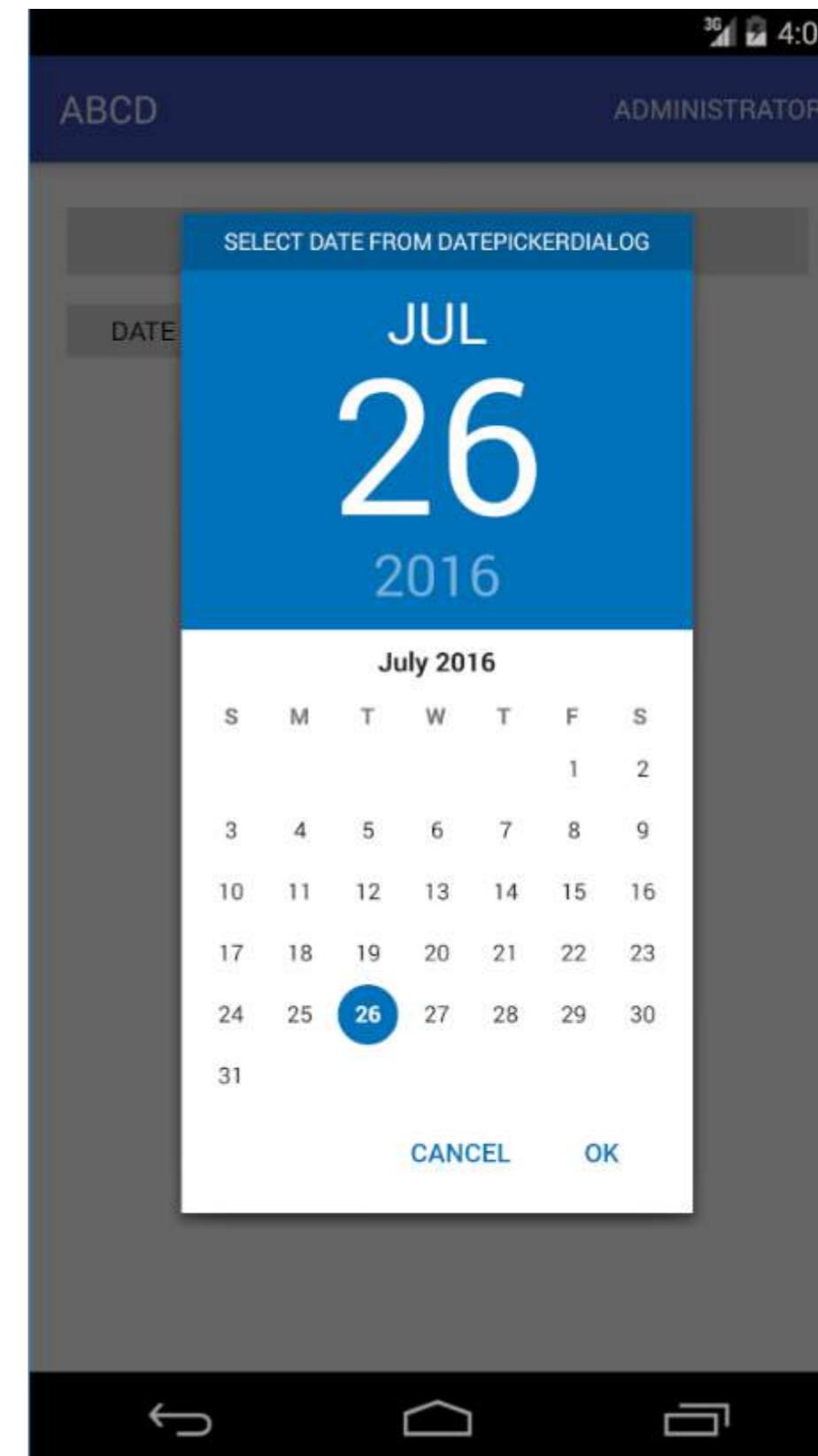
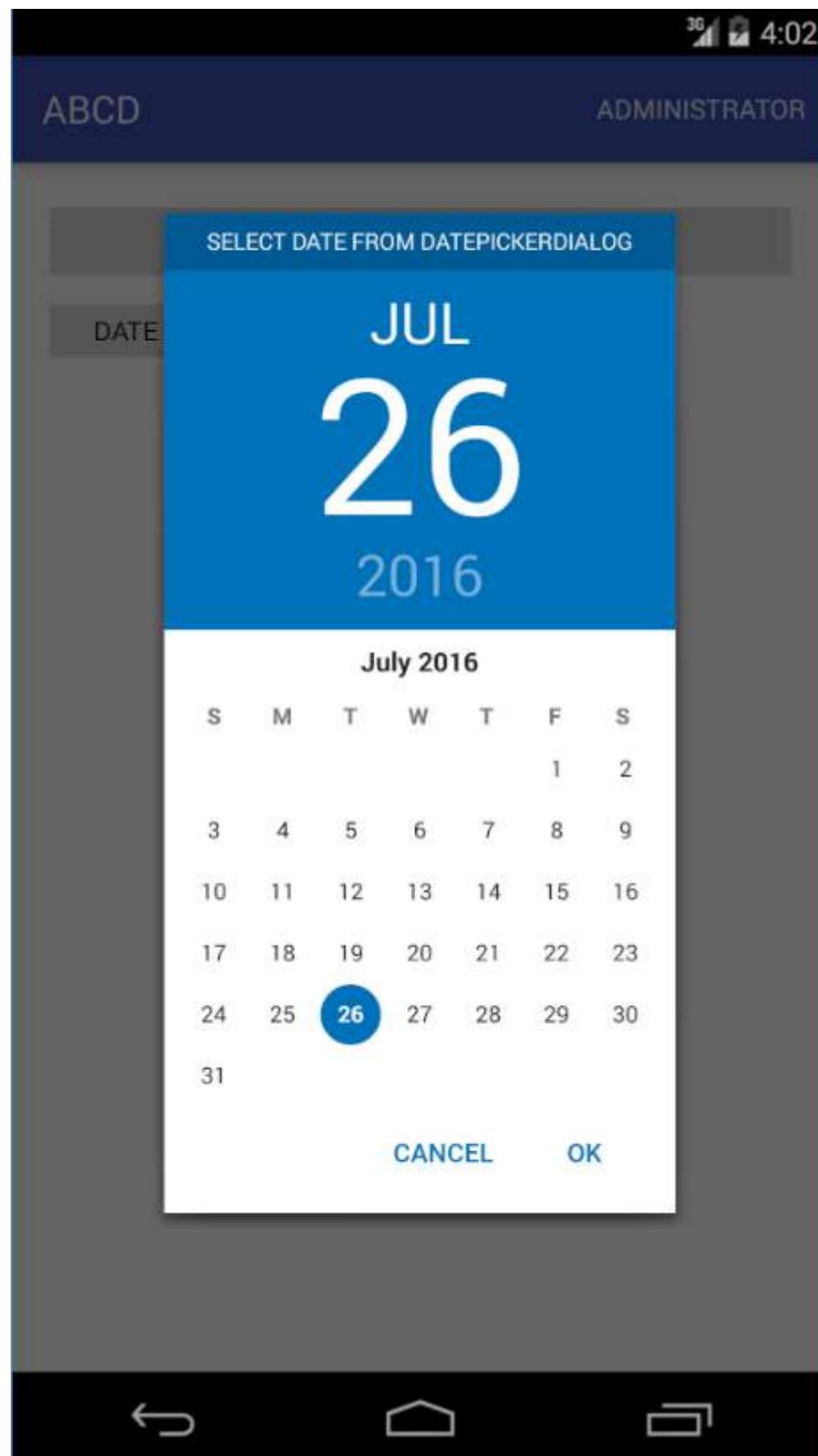
    String date = "Selected Date : " + Day + "-" + Month + "-" + Year;

    Toast.makeText(MainActivity.this, date, Toast.LENGTH_LONG).show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    getMenuInflater().inflate(R.menu.abc_main_menu, menu);
    return true;
}
}

```

Output:



第100章：Android中的本地化日期/时间

第100.1节：使用

DateUtils.formatDateTime()自定义本地化日期格式

DateUtils.formatDateTime()允许你提供一个时间，根据你提供的标志，创建一个本地化的日期时间字符串。标志允许你指定是否包含特定元素（如星期几）。

```
Date date = new Date(); String localizedDate = DateUtils.formatDateTime(context, date.getTime(),  
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTime() 会自动处理正确的日期格式。

第100.2节：Android中的标准日期/时间格式

格式化日期：

```
Date date = new Date(); DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM); String localizedDate =  
df.format(date)
```

格式化日期和时间。日期为短格式，时间为长格式：

```
Date date = new Date(); DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);  
String localizedDate = df.format(date)
```

第100.3节：完全自定义日期/时间

```
Date date = new Date(); df = new SimpleDateFormat("HH:mm", Locale.US); String localizedDate = df.format(date)
```

常用模式：

- HH : 小时 (0-23)
- hh : 小时 (1-12)
- a : 上午/下午标记
- mm : 分钟 (0-59)
- ss : 秒
- dd : 月份中的天 (1-31)
- MM : 月份
- yyyy : 年份

Chapter 100: Localized Date/Time in Android

Section 100.1: Custom localized date format with DateUtils.formatDateTime()

[DateUtils.formatDateTime\(\)](#) allows you to supply a time, and based on the flags you provide, it creates a localized datetime string. The flags allow you to specify whether to include specific elements (like the weekday).

```
Date date = new Date(); String localizedDate = DateUtils.formatDateTime(context, date.getTime(),  
DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);
```

formatDateTime() automatically takes care about proper date formats.

Section 100.2: Standard date/time formatting in Android

Format a date:

```
Date date = new Date(); DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM); String localizedDate =  
df.format(date)
```

Format a date and time. Date is in short format, time is in long format:

```
Date date = new Date(); DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG);  
String localizedDate = df.format(date)
```

Section 100.3: Fully customized date/time

```
Date date = new Date(); df = new SimpleDateFormat("HH:mm", Locale.US); String localizedDate = df.format(date)
```

Commonly used patterns:

- HH: hour (0-23)
- hh: hour (1-12)
- a: AM/PM marker
- mm: minute (0-59)
- ss: second
- dd: day in month (1-31)
- MM: month
- yyyy: year

第101章：时间工具

第101.1节：检查是否在某个时间段内

此示例将帮助验证给定时间是否在某个时间段内。

要检查时间是否是今天，我们可以使用 `DateUtils` 类

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

要检查时间是否在一周内，

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

要检查时间是否在一年内，

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

要检查时间是否在包括今天在内的若干天内，

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

注：`DateUtils` 是 `android.text.format.DateUtils`

第101.2节：将日期格式转换为毫秒

要将日期从 `dd/MM/yyyy` 格式转换为毫秒，您可以调用此函数，参数为字符串格式的日期

```
public long getMilliFromDate(String dateFormat) {
    Date date = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    try {
date = formatter.parse(dateFormat);
    } catch (ParseException e) {
e.printStackTrace();
    }
    System.out.println("Today is " + date);
    return date.getTime();
}
```

此方法将毫秒转换为时间戳格式的日期：

```
public String getTimeStamp(long timeInMillis) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); // 修改格式
}
```

Chapter 101: Time Utils

Section 101.1: To check within a period

This example will help to verify the given time is within a period or not.

To check the time is today, We can use `DateUtils` class

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

To check the time is within a week,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

To check the time is within a year,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

To check the time is within a number day of day including today,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Note : `DateUtils` is `android.text.format.DateUtils`

Section 101.2: Convert Date Format into Milliseconds

To Convert you date in `dd/MM/yyyy` format into milliseconds you call this function with data as String

```
public long getMilliFromDate(String dateFormat) {
    Date date = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    try {
date = formatter.parse(dateFormat);
    } catch (ParseException e) {
e.printStackTrace();
    }
    System.out.println("Today is " + date);
    return date.getTime();
}
```

This method converts milliseconds to Time-stamp Format date :

```
public String getTimeStamp(long timeInMillis) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); // modify format
}
```

```

date = formatter.format(new Date(timeInMillies));
System.out.println("Today is " + date);

return date;
}

```

此方法将把给定的具体日、月、年转换为毫秒。在使用
Timpicker 或 Datepicker 时非常有用

```

public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}

```

它将返回日期对应的毫秒数

```

public static String getNormalDate(long timeInMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = formatter.format(timeInMillies);
    System.out.println("Today is " + date);
    return date;
}

```

它将返回当前日期

```

public static String getCurrentDate() {
    Calendar c = Calendar.getInstance();
    System.out.println("Current time => " + c.getTime());
    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    String formattedDate = df.format(c.getTime());
    return formattedDate;
}

```

注意：Java 提供多种日期格式支持 日期模式

第101.3节：获取当前真实时间

此方法计算当前设备时间并加减真实时间与设备时间之间的差值

```

public static Calendar getCurrentRealTime() {

    long bootTime = networkTime - SystemClock.elapsedRealtime();
    Calendar calInstance = Calendar.getInstance();
    calInstance.setTimeZone(getUTCTimeZone());
    long currentDeviceTime = bootTime + SystemClock.elapsedRealtime();
    calInstance.setInMillis(currentDeviceTime);
    return calInstance;
}

```

获取基于UTC的时区。

```

public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}

```

```

date = formatter.format(new Date(timeInMillies));
System.out.println("Today is " + date);

return date;
}

```

This Method will convert given specific day,month and year into milliseconds. It will be very help when using
Timpicker or Datepicker

```

public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}

```

It will return milliseconds from date

```

public static String getNormalDate(long timeInMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = formatter.format(timeInMillies);
    System.out.println("Today is " + date);
    return date;
}

```

It will return current date

```

public static String getCurrentDate() {
    Calendar c = Calendar.getInstance();
    System.out.println("Current time => " + c.getTime());
    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    String formattedDate = df.format(c.getTime());
    return formattedDate;
}

```

Note : Java Provides numbers of date format support [Date Pattern](#)

Section 101.3: GetCurrentRealTime

This calculate current device time and add/subtract difference between real and device time

```

public static Calendar getCurrentRealTime() {

    long bootTime = networkTime - SystemClock.elapsedRealtime();
    Calendar calInstance = Calendar.getInstance();
    calInstance.setTimeZone(getUTCTimeZone());
    long currentDeviceTime = bootTime + SystemClock.elapsedRealtime();
    calInstance.setInMillis(currentDeviceTime);
    return calInstance;
}

```

get UTC based timezone.

```

public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}

```

第102章：应用内计费

第102.1节：可消耗的应用内购买

可消耗的托管产品是指可以多次购买的产品，例如游戏内货币、游戏生命、增强道具等。

在本例中，我们将实现4种不同的可消耗托管产品 "item1"、"item2"、"item3"、"item4"。

步骤总结：

1. 将应用内购买库添加到您的项目中（AIDL 文件）。
2. 在AndroidManifest.xml文件中添加所需权限。
3. 将签名的apk部署到Google开发者控制台。
4. 定义您的产品。
5. 实现代码。
6. 测试应用内购买（可选）。

步骤 1：

首先，我们需要将AIDL文件添加到您的项目中，Google文档中对此有明确说明 [here](#)。

`IInAppBillingService.aidl` 是一个Android接口定义语言（AIDL）文件，定义了应用内购买版本3服务的接口。您将使用此接口通过调用IPC方法来发起计费请求。

步骤 2：

添加AIDL文件后，在AndroidManifest.xml中添加BILLING权限：

```
<!-- 实现应用内购买所需的权限 -->
<uses-permission android:name="com.android.vending.BILLING" />
```

步骤3：

生成已签名的apk，并上传到Google开发者控制台。这是必须的，以便我们可以开始在那里定义我们的应用内产品。

步骤4：

为所有产品定义不同的productID，并为每个产品设置价格。产品有两种类型（管理型产品和订阅）。如前所述，我们将实现4个不同的可消耗管理型产品 "item1"、"item2"、"item3"、"item4"。

步骤5：

完成上述所有步骤后，您现在可以开始在自己的Activity中实现代码了。

MainActivity：

```
public class MainActivity extends Activity {

    IInAppBillingService inAppBillingService;
    ServiceConnection serviceConnection;
```

Chapter 102: In-app Billing

Section 102.1: Consumable In-app Purchases

Consumable Managed Products are products that can be bought multiple times such as in-game currency, game lives, power-ups, etc.

In this example, we are going to implement 4 different consumable **managed products** "item1", "item2", "item3", "item4".

Steps in summary:

1. Add the In-app Billing library to your project (AIDL File).
2. Add the required permission in `AndroidManifest.xml` file.
3. Deploy a signed apk to Google Developers Console.
4. Define your products.
5. Implement the code.
6. Test In-app Billing (optional).

Step 1:

First of all, we will need to add the AIDL file to your project as clearly explained in Google Documentation [here](#).

`IInAppBillingService.aidl` is an Android Interface Definition Language (AIDL) file that defines the interface to the In-app Billing Version 3 service. You will use this interface to make billing requests by invoking IPC method calls.

Step 2:

After adding the AIDL file, add BILLING permission in `AndroidManifest.xml`:

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Step 3:

Generate a signed apk, and upload it to Google Developers Console. This is required so that we can start defining our in-app products there.

Step 4:

Define all your products with different productID, and set a price to each one of them. There are 2 types of products (Managed Products and Subscriptions). As we already said, we are going to implement 4 different consumable **managed products** "item1", "item2", "item3", "item4".

Step 5:

After doing all the steps above, you are now ready to start implementing the code itself in your own activity.

MainActivity:

```
public class MainActivity extends Activity {

    IInAppBillingService inAppBillingService;
    ServiceConnection serviceConnection;
```

```

// 每个商品的productID。您应在Google开发者控制台中定义它们。
final String item1 = "item1";
final String item2 = "item2";
final String item3 = "item3";
final String item4 = "item4";

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // 根据您的布局文件实例化视图。
    final Button buy1 = (Button) findViewById(R.id.buy1);
    final Button buy2 = (Button) findViewById(R.id.buy2);
    final Button buy3 = (Button) findViewById(R.id.buy3);
    final Button buy4 = (Button) findViewById(R.id.buy4);

    // 为每个按钮设置 setOnClickListener()。
    // buyItem() 是我们将实现的用于启动购买流程的方法。
    buy1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item1);
        }
    });

    buy2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item2);
        }
    });

    buy3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item3);
        }
    });

    buy4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item4);
        }
    });

    // 绑定服务连接。
    serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceDisconnected(ComponentName name) {
            inAppBillingService = null;
        }

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            inAppBillingService = IInAppBillingService.Stub.asInterface(service);
        }
    };

    // 绑定服务。
    Intent serviceIntent = new Intent("com.android.vending.billing.InAppBillingService.BIND");

```

```

// productID for each item. You should define them in the Google Developers Console.
final String item1 = "item1";
final String item2 = "item2";
final String item3 = "item3";
final String item4 = "item4";

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Instantiate the views according to your layout file.
    final Button buy1 = (Button) findViewById(R.id.buy1);
    final Button buy2 = (Button) findViewById(R.id.buy2);
    final Button buy3 = (Button) findViewById(R.id.buy3);
    final Button buy4 = (Button) findViewById(R.id.buy4);

    // setOnClickListener() for each button.
    // buyItem() here is the method that we will implement to launch the PurchaseFlow.
    buy1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item1);
        }
    });

    buy2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item2);
        }
    });

    buy3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item3);
        }
    });

    buy4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            buyItem(item4);
        }
    });

    // Attach the service connection.
    serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceDisconnected(ComponentName name) {
            inAppBillingService = null;
        }

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            inAppBillingService = IInAppBillingService.Stub.asInterface(service);
        }
    };

    // Bind the service.
    Intent serviceIntent = new Intent("com.android.vending.billing.InAppBillingService.BIND");

```

```

serviceIntent.setPackage("com.android.vending");
bindService(serviceIntent, serviceConnection, BIND_AUTO_CREATE);

// 获取每个产品的价格，并将价格设置为文本显示在
// 每个按钮上，以便用户知道每个商品的价格。
if (inAppBillingService != null) {
    // 注意：这里需要创建一个新线程，因为
    // getSkuDetails() 会触发网络请求，
    // 如果在主线程调用，可能会导致应用卡顿。
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
ArrayList<String> skuList = new ArrayList<>();
skuList.add(item1);
skuList.add(item2);
skuList.add(item3);
skuList.add(item4);
Bundle querySkus = new Bundle();
querySkus.putStringArrayList("ITEM_ID_LIST", skuList);

try {
Bundle skuDetails = inAppBillingService.getSkuDetails(3, getPackageName(),
"inapp", querySkus);
int response = skuDetails.getInt("RESPONSE_CODE");

if (response == 0) {
ArrayList<String> responseList =
skuDetails.getStringArrayList("DETAILS_LIST");

for (String thisResponse : responseList) {
JSONObject object = new JSONObject(thisResponse);
String sku = object.getString("productId");
String price = object.getString("price");

switch (sku) {
case item1:
buy1.setText(price);
break;
case item2:
buy2.setText(price);
break;
case item3:
buy3.setText(price);
break;
case item4:
buy4.setText(price);
break;
}
}
} catch (RemoteException | JSONException e) {
e.printStackTrace();
}
});
线程启动();
}

// 启动购买流程，传入用户想购买的商品的 productId 作为
// 参数。
private void 购买商品(String productId) {

```

```

serviceIntent.setPackage("com.android.vending");
bindService(serviceIntent, serviceConnection, BIND_AUTO_CREATE);

// Get the price of each product, and set the price as text to
// each button so that the user knows the price of each item.
if (inAppBillingService != null) {
    // Attention: You need to create a new thread here because
    // getSkuDetails() triggers a network request, which can
    // cause lag to your app if it was called from the main thread.
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
ArrayList<String> skuList = new ArrayList<>();
skuList.add(item1);
skuList.add(item2);
skuList.add(item3);
skuList.add(item4);
Bundle querySkus = new Bundle();
querySkus.putStringArrayList("ITEM_ID_LIST", skuList);

try {
Bundle skuDetails = inAppBillingService.getSkuDetails(3, getPackageName(),
"inapp", querySkus);
int response = skuDetails.getInt("RESPONSE_CODE");

if (response == 0) {
ArrayList<String> responseList =
skuDetails.getStringArrayList("DETAILS_LIST");

for (String thisResponse : responseList) {
JSONObject object = new JSONObject(thisResponse);
String sku = object.getString("productId");
String price = object.getString("price");

switch (sku) {
case item1:
buy1.setText(price);
break;
case item2:
buy2.setText(price);
break;
case item3:
buy3.setText(price);
break;
case item4:
buy4.setText(price);
break;
}
}
} catch (RemoteException | JSONException e) {
e.printStackTrace();
}
});
thread.start();
}

// Launch the PurchaseFlow passing the productId of the item the user wants to buy as a
// parameter.
private void buyItem(String productId) {

```

```

if (内购服务 != null) {
    尝试 {
        Bundle 购买意图包 = 内购服务.获取购买意图(3, 获取包名(),
        productID, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
        PendingIntent 待处理意图 = 购买意图包.获取可序列化对象("BUY_INTENT");
        启动意图发送者以获取结果(待处理意图.获取意图发送者(), 1003, 新 Intent(), 0,
        0, 0);
    } 捕获 (RemoteException | IntentSender.发送意图异常 e) {
        e.printStackTrace();
    }
}

// 在 onDestroy() 中解绑服务。如果你不解绑，打开的
// 服务连接可能会导致设备性能下降。
@Override
public void onDestroy() {
    super.onDestroy();
    if (inAppBillingService != null) {
        unbindService(serviceConnection);
    }
}

// 在这里检查应用内购买是否成功。如果成功，
// 则消耗该商品，并让应用做出相应的更改。
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == 1003 && resultCode == RESULT_OK) {

        final String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");

        // 注意：这里需要创建一个新线程，因为
        // consumePurchase() 会触发网络请求，
        // 如果在主线程调用会导致应用卡顿。
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    JSONObject jo = new JSONObject(purchaseData);
                    // 获取已购买商品的 productID。
                    String sku = jo.getString("productId");
                    String productName = null;

                    // increaseCoins() 是游戏中用作示例的方法，
                    // 用于在购买成功后增加游戏内货币。
                    // 你应该在这里实现自己的代码，
                    // 并让应用在购买成功后应用所需的更改。
                    switch (sku) {
                        case item1:
                            productName = "物品 1";
                            increaseCoins(2000);
                            break;
                        case item2:
                            productName = "物品 2";
                            increaseCoins(8000);
                            break;
                        case item3:
                            productName = "物品 3";
                            increaseCoins(18000);
                            break;
                    }
                }
            }
        });
        thread.start();
    }
}

```

```

if (inAppBillingService != null) {
    try {
        Bundle buyIntentBundle = inAppBillingService.getBuyIntent(3, getPackageName(),
        productID, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
        PendingIntent pendingIntent = buyIntentBundle.getParcelable("BUY_INTENT");
        startIntentSenderForResult(pendingIntent.getIntentSender(), 1003, new Intent(), 0,
        0, 0);
    } catch (RemoteException | IntentSender.SendIntentException e) {
        e.printStackTrace();
    }
}

// Unbind the service in onDestroy(). If you don't unbind, the open
// service connection could cause your device's performance to degrade.
@Override
public void onDestroy() {
    super.onDestroy();
    if (inAppBillingService != null) {
        unbindService(serviceConnection);
    }
}

// Check here if the in-app purchase was successful or not. If it was successful,
// then consume the product, and let the app make the required changes.
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == 1003 && resultCode == RESULT_OK) {

        final String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");

        // Attention: You need to create a new thread here because
        // consumePurchase() triggers a network request, which can
        // cause lag to your app if it was called from the main thread.
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    JSONObject jo = new JSONObject(purchaseData);
                    // Get the productID of the purchased item.
                    String sku = jo.getString("productId");
                    String productName = null;

                    // increaseCoins() here is a method used as an example in a game to
                    // increase the in-game currency if the purchase was successful.
                    // You should implement your own code here, and let the app apply
                    // the required changes after the purchase was successful.
                    switch (sku) {
                        case item1:
                            productName = "Item 1";
                            increaseCoins(2000);
                            break;
                        case item2:
                            productName = "Item 2";
                            increaseCoins(8000);
                            break;
                        case item3:
                            productName = "Item 3";
                            increaseCoins(18000);
                            break;
                    }
                }
            }
        });
        thread.start();
    }
}

```

```

        case item4:
            productName = "物品 4";
            increaseCoins(30000);
            break;
    }

    // 消耗购买，以便用户能够再次购买相同的
    产品。
    inAppBillingService.consumePurchase(3, getPackageName(),
        jo.getString("purchaseToken"));
    Toast.makeText(MainActivity.this, productName + " 已成功
    购买。绝佳的选择，主人！", Toast.LENGTH_LONG).show();
} catch (JSONException | RemoteException e) {
    Toast.makeText(MainActivity.this, "解析购买数据失败。",
        Toast.LENGTH_LONG).show();
    e.printStackTrace();
}
}

线程启动();
}
}

```

步骤 6：

实现代码后，您可以通过将 apk 部署到测试版/内测渠道来测试，并让其他用户帮您测试代码。然而，在测试模式下无法进行真实的应用内购买。您必须先将应用/游戏发布到 Play 商店，所有产品才能完全激活。

有关测试应用内计费的更多信息，请参见 [here](#)。

第 102.2 节：（第三方）应用内 v3 库

步骤1：首先按照以下两个步骤添加应用功能：

1. 使用以下方式添加库：

```

repositories {
    mavenCentral()
}
dependencies {
compile 'com.anjlab.android.iab.v3:library:1.0.+'
}

```

2. 在清单文件中添加权限。

```
<uses-permission android:name="com.android.vending.BILLING" />
```

步骤 2：初始化您的计费处理器：

```
BillingProcessor bp = new BillingProcessor(this, "请在此处填写您的 Google Play 控制台许可证密钥",
this);
```

并实现计费处理器接口：BillingProcessor.IBillingHandler，其中包含4个方法：a. onBillingInitialized();
b. onProductPurchased(String productId, TransactionDetails details)：在此处理成功购买后的操作
c. onBillingError(int errorCode, Throwable error)：处理购买过程中发生的任何错误
d. onPurchaseHistoryRestored()：用于恢复应用内购买记录

```

        case item4:
            productName = "Item 4";
            increaseCoins(30000);
            break;
    }

    // Consume the purchase so that the user is able to purchase the same
    product again.
    inAppBillingService.consumePurchase(3, getPackageName(),
        jo.getString("purchaseToken"));
    Toast.makeText(MainActivity.this, productName + " is successfully
    purchased. Excellent choice, master!", Toast.LENGTH_LONG).show();
} catch (JSONException | RemoteException e) {
    Toast.makeText(MainActivity.this, "Failed to parse purchase data.",
        Toast.LENGTH_LONG).show();
    e.printStackTrace();
}
}

thread.start();
}
}

```

Step 6:

After implementing the code, you can test it by deploying your apk to beta/alpha channel, and let other users test the code for you. However, real in-app purchases can't be made while in testing mode. You have to publish your app/game first to Play Store so that all the products are fully activated.

More info on testing In-app Billing can be found [here](#).

Section 102.2: (Third party) In-App v3 Library

Step 1: First of all follow these two steps to add in app functionality :

1. Add the library using :

```

repositories {
    mavenCentral()
}
dependencies {
compile 'com.anjlab.android.iab.v3:library:1.0.+'
}

```

2. Add permission in manifest file.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Step 2: Initialise your billing processor:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE HERE",
this);
```

and implement Billing Handler : BillingProcessor.IBillingHandler which contains 4 methods : a. onBillingInitialized();
b. onProductPurchased(String productId, TransactionDetails details) : This is where you need to handle actions to
be performed after successful purchase c. onBillingError(int errorCode, Throwable error) : Handle any error
occurred during purchase process d. onPurchaseHistoryRestored() : For restoring in app purchases

步骤 3：如何购买产品。

购买托管产品：

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

购买订阅：

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

步骤4：消费产品。

要消费产品，只需调用consumePurchase方法。

```
bp.consumePurchase("YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

有关应用内其他方法，请访问[github](#)

Step 3: How to purchase a product.

To purchase a managed product :

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

And to Purchase a subscription :

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Step 4 : Consuming a product.

To consume a product simply call consumePurchase method.

```
bp.consumePurchase("YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

For other methods related to in app visit [github](#)

第103章：FloatingActionButton

参数

	详情
android.support.design:elevation	FAB 的阴影值。可以是对另一个资源的引用，形式为 "@[+][package:]type/name"，或者是主题属性，形式为 "?[package:]type/name"。
android.support.design:fabSize	FAB 的大小。
android.support.design:rippleColor	FAB 的波纹颜色。
android.support.design:useCompatPadding	启用兼容内边距。

浮动操作按钮用于一种特殊类型的突出操作，默认情况下，它以扩展的材质块动画形式出现在屏幕上。按钮内的图标也可以动画显示，且 FAB 的移动方式可能与其他 UI 元素不同，因为它们的重要性不同。浮动操作按钮代表应用中的主要操作，可以简单地触发一个动作或导航到某处。

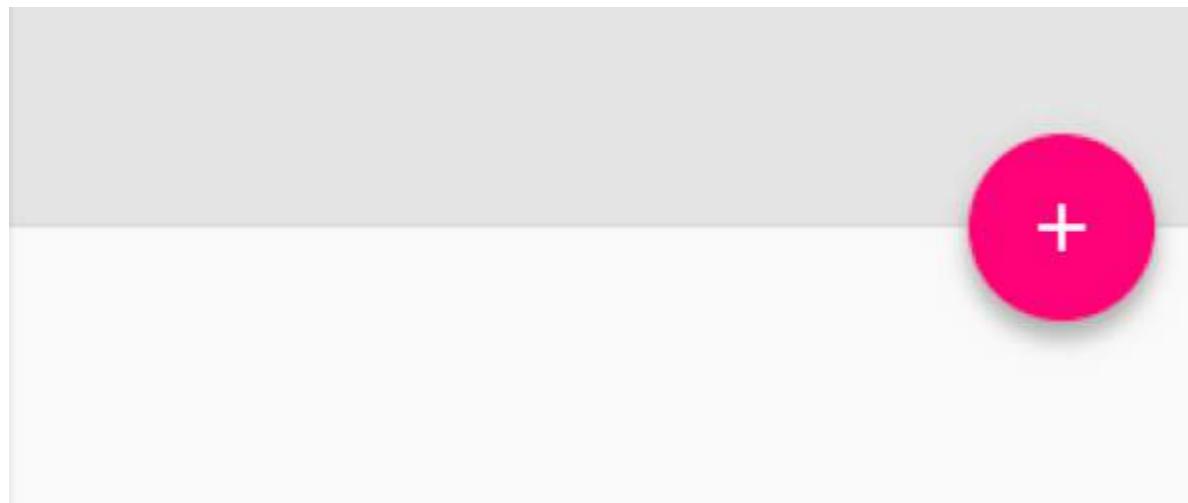
第 103.1 节：如何将 FAB 添加到布局中

要使用 FloatingActionButton，只需按照备注部分的说明，在 build.gradle 文件中添加依赖即可。

然后添加到布局中：

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@drawable/my_icon" />
```

示例：



颜色

此视图的背景颜色默认为您的主题的 colorAccent 颜色。

在上图中，如果 src 仅指向 + 图标（默认24x24 dp），要获得完整圆形的背景颜色，您可以使用 app:backgroundTint="@color/your_colour"

如果您想在代码中更改颜色，可以使用，

Chapter 103: FloatingActionButton

Parameter

	Detail
android.support.design:elevation	Elevation value for the FAB. May be a reference to another resource, in the form "@[+][package:]type/name" or a theme attribute in the form "?[package:]type/name".
android.support.design:fabSize	Size for the FAB.
android.support.design:rippleColor	Ripple color for the FAB.
android.support.design:useCompatPadding	Enable compat padding.

Floating action button is used for a special type of promoted action, it animates onto the screen as an expanding piece of material, by default. The icon within it may be animated, also FAB may move differently than other UI elements because of their relative importance. A floating action button represents the primary action in an application which can simply trigger an action or navigate somewhere.

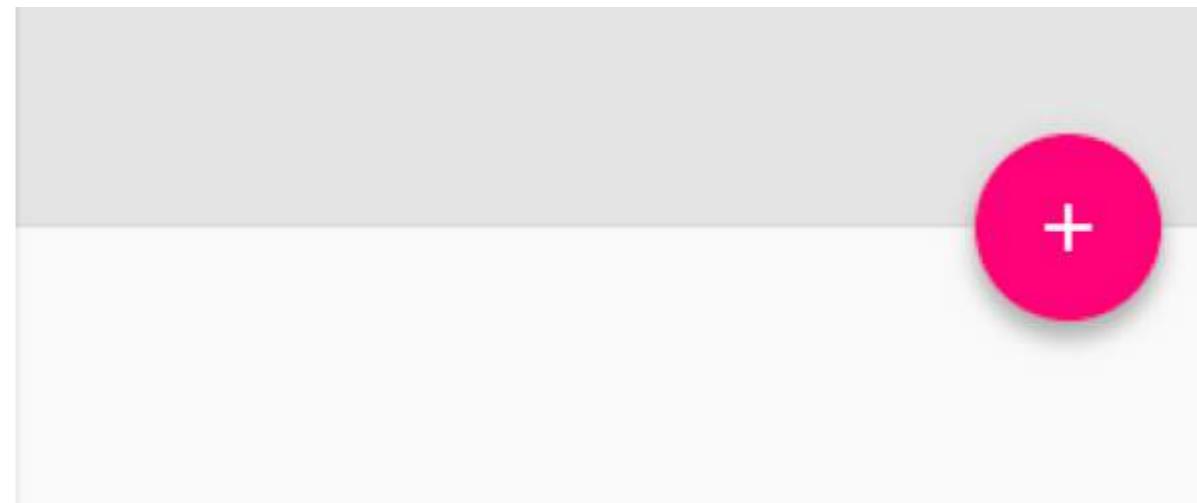
Section 103.1: How to add the FAB to the layout

To use a FloatingActionButton just add the dependency in the build.gradle file as described in the remarks section.

Then add to the layout:

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@drawable/my_icon" />
```

An example:



Color

The background color of this view defaults to the your theme's colorAccent.

In the above image if the src only points to + icon (by default 24x24 dp), to get the background color of full circle you can use app:backgroundTint="@color/your_colour"

If you wish to change the color in code you can use,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(您的int颜色));
```

如果您想更改 FAB 按下状态时的颜色, 请使用

```
mFab.setRippleColor(你的颜色, 类型为int);
```

定位

建议在移动设备上距离边缘至少留16dp, 在平板/桌面设备上至少留24dp。

注意: 一旦设置了除覆盖整个FloatingActionButton区域外的src, 请确保该图像的尺寸正确, 以获得最佳效果。

默认圆形尺寸为56 x 56dp



迷你圆形尺寸 : 40 x 40dp

如果只想更改内部图标, 默认尺寸请使用24 x 24dp的图标

第103.2节 : 滑动时显示和隐藏FloatingActionButton

要使用默认动画显示和隐藏FloatingActionButton, 只需调用方法show()和hide()。建议将FloatingActionButton保留在Activity布局中, 而不是放在Fragment中, 这样可以使显示和隐藏时的默认动画正常工作。

以下是一个使用ViewPager的示例:

- 三个标签页
- 显示FloatingActionButton在第一个和第三个标签页
- 隐藏中间标签页的FloatingActionButton

```
public class MainActivity extends AppCompatActivity {
    FloatingActionButton fab;
    ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fab = (FloatingActionButton) findViewById(R.id.fab);
        viewPager = (ViewPager) findViewById(R.id.viewpager);

        // ..... 设置 ViewPager .....
    }

    viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
        @Override
        public void onPageSelected(int position) {
            if (position == 0) {
                fab.setImageResource(android.R.drawable.ic_dialog_email);
                fab.show();
            } else if (position == 2) {
        }
    })
}
```

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

If you want to change FAB's color in pressed state use

```
mFab.setRippleColor(your color in int);
```

Positioning

It is recommended to place 16dp minimum from the edge on mobile, and 24dp minimum on tablet/desktop.

Note : Once you set an src excepting to cover the full area of FloatingActionButton make sure you have the right size of that image to get the best result.

Default circle size is 56 x 56dp



Mini circle size : 40 x 40dp

If you only want to change only the Interior icon use a 24 x 24dp icon for default size

Section 103.2: Show and Hide FloatingActionButton on Swipe

To show and hide a FloatingActionButton with the default animation, just call the methods show() and hide(). It's good practice to keep a FloatingActionButton in the Activity layout instead of putting it in a Fragment, this allows the default animations to work when showing and hiding.

Here is an example with a ViewPager:

- Three Tabs
- Show FloatingActionButton for the first and third Tab
- Hide the FloatingActionButton on the middle Tab

```
public class MainActivity extends AppCompatActivity {
    FloatingActionButton fab;
    ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fab = (FloatingActionButton) findViewById(R.id.fab);
        viewPager = (ViewPager) findViewById(R.id.viewpager);

        // ..... set up ViewPager .....

        viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageSelected(int position) {
                if (position == 0) {
                    fab.setImageResource(android.R.drawable.ic_dialog_email);
                    fab.show();
                } else if (position == 2) {
                    fab.setImageResource(android.R.drawable.ic_dialog_email);
                    fab.show();
                } else if (position == 1) {
                    fab.setVisibility(View.GONE);
                }
            }
        });
    }
}
```

```

fab.setImageResource(android.R.drawable.ic_dialog_map);
    fab.show();
} else {
fab.hide();
}

@Override
public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {}

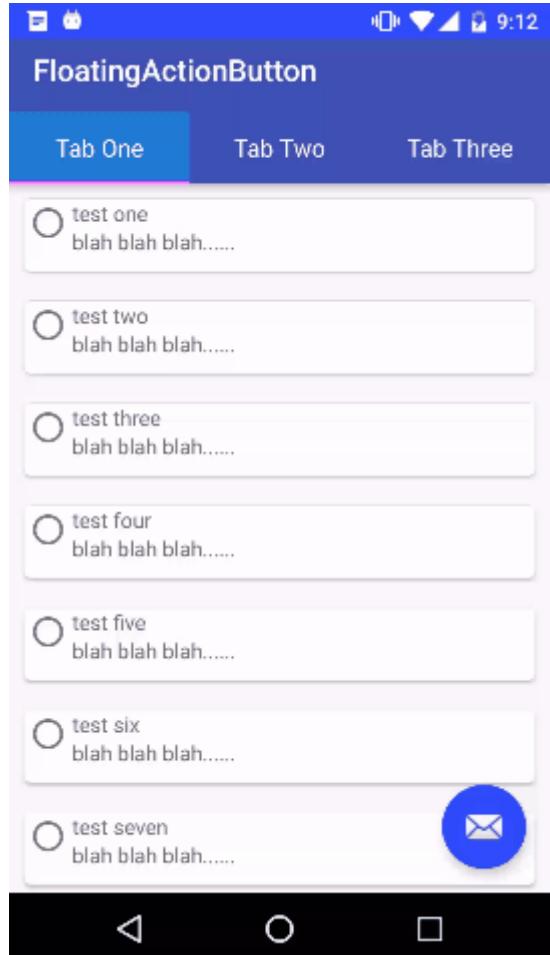
@Override
public void onPageScrollStateChanged(int state) {}

// 处理 FloatingActionButton 的点击事件：
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int position = viewPager.getCurrentItem();
        if (position == 0) {
openSend();
    } else if (position == 2) {
        openMap();
    }
}
});

}
}

```

结果：



```

fab.setImageResource(android.R.drawable.ic_dialog_map);
    fab.show();
} else {
    fab.hide();
}

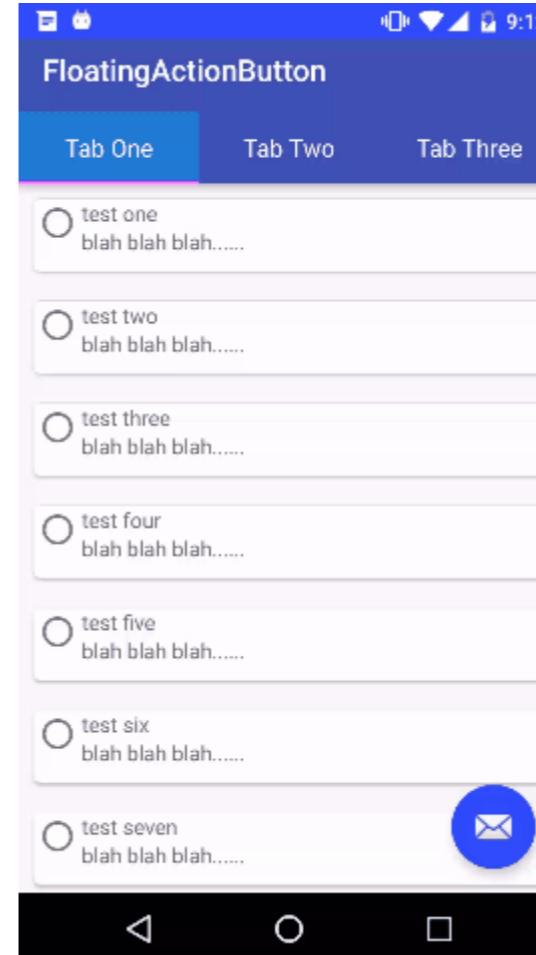
@Override
public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {}

@Override
public void onPageScrollStateChanged(int state) {}

// Handle the FloatingActionButton click event:
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int position = viewPager.getCurrentItem();
        if (position == 0) {
openSend();
    } else if (position == 2) {
        openMap();
    }
}
});
}

```

Result:



第103.3节：滚动时显示和隐藏 FloatingActionButton

从支持库版本22.2.1开始，可以通过使用 `FloatingActionButton.Behavior` 子类来实现滚动时显示和隐藏 `FloatingActionButton`，该子类利用了 `show()` 和 `hide()`方法。

请注意，这仅适用于带有`CoordinatorLayout`且内部视图支持嵌套滚动的情况，例如`RecyclerView`和`NestedScrollView`。

这个`ScrollAwareFABBehavior`类来自Android Guides on Codepath (cc-wiki, 需注明出处)

```
public class ScrollAwareFABBehavior extends FloatingActionButton.Behavior {
    public ScrollAwareFABBehavior(Context context, AttributeSet attrs) {
        super();
    }

    @Override
    public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, FloatingActionButton child,
                                       View directTargetChild, View target, int nestedScrollAxes) {
        // 确保我们响应垂直滚动
        return nestedScrollAxes == ViewCompat.SCROLL_AXIS_VERTICAL
            || super.onStartNestedScroll(coordinatorLayout, child, directTargetChild, target, nestedScrollAxes);
    }

    @Override
    public void onNestedScroll(CoordinatorLayout coordinatorLayout, FloatingActionButton child,
                               View target, int dxConsumed, int dyConsumed,
                               int dxUnconsumed, int dyUnconsumed) {
        super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed,
                           dxUnconsumed, dyUnconsumed);
        if (dyConsumed > 0 && child.getVisibility() == View.VISIBLE) {
            // 用户向下滚动且悬浮按钮当前可见 -> 隐藏悬浮按钮
            child.hide();
        } else if (dyConsumed < 0 && child.getVisibility() != View.VISIBLE) {
            // 用户向上滚动且悬浮按钮当前不可见 -> 显示悬浮按钮
            child.show();
        }
    }
}
```

在 `FloatingActionButton` 布局 xml 中，指定 `app:layout_behavior` 为完整类名 `ScrollAwareFABBehavior`：

```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

例如使用以下布局：

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

Section 103.3: Show and Hide FloatingActionButton on Scroll

Starting with the Support Library version 22.2.1, it's possible to show and hide a `FloatingActionButton` from scrolling behavior using a `FloatingActionButton.Behavior` subclass that takes advantage of the `show()` and `hide()` methods.

Note that this only works with a `CoordinatorLayout` in conjunction with inner Views that support Nested Scrolling, such as `RecyclerView` and `NestedScrollView`.

This `ScrollAwareFABBehavior` class comes from the [Android Guides on Codepath](#) (cc-wiki with attribution required)

```
public class ScrollAwareFABBehavior extends FloatingActionButton.Behavior {
    public ScrollAwareFABBehavior(Context context, AttributeSet attrs) {
        super();
    }

    @Override
    public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, FloatingActionButton child,
                                       View directTargetChild, View target, int nestedScrollAxes) {
        // Ensure we react to vertical scrolling
        return nestedScrollAxes == ViewCompat.SCROLL_AXIS_VERTICAL
            || super.onStartNestedScroll(coordinatorLayout, child, directTargetChild, target, nestedScrollAxes);
    }

    @Override
    public void onNestedScroll(CoordinatorLayout coordinatorLayout, FloatingActionButton child,
                               View target, int dxConsumed, int dyConsumed,
                               int dxUnconsumed, int dyUnconsumed) {
        super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed,
                           dxUnconsumed, dyUnconsumed);
        if (dyConsumed > 0 && child.getVisibility() == View.VISIBLE) {
            // User scrolled down and the FAB is currently visible -> hide the FAB
            child.hide();
        } else if (dyConsumed < 0 && child.getVisibility() != View.VISIBLE) {
            // User scrolled up and the FAB is currently not visible -> show the FAB
            child.show();
        }
    }
}
```

In the `FloatingActionButton` layout xml, specify the `app:layout_behavior` with the fully-qualified-class-name of `ScrollAwareFABBehavior`:

```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

For example with this layout:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```

<android.support.design.widget.AppBarLayout
    android:id="@+id/appBarLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:elevation="6dp">
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:elevation="0dp"
        app:layout_scrollFlags="scroll|enterAlways"
    />

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
    />

</android.support.design.widget.AppBarLayout>

<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_below="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
/>

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
    android:layout_margin="@dimen/fab_margin"
    android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>

```

结果如下：

```

<android.support.design.widget.AppBarLayout
    android:id="@+id/appBarLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:elevation="6dp">
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:elevation="0dp"
        app:layout_scrollFlags="scroll|enterAlways"
    />

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
    />

</android.support.design.widget.AppBarLayout>

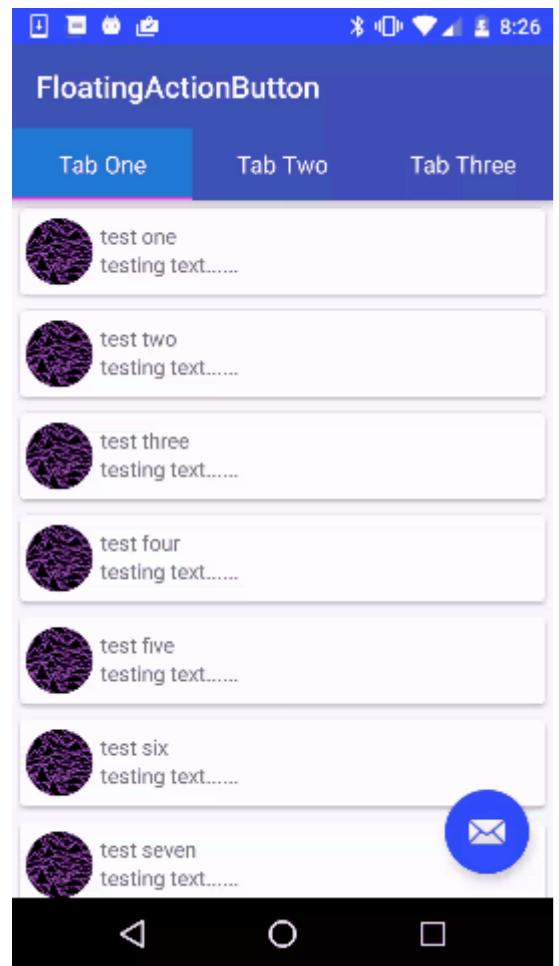
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_below="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
/>

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
    android:layout_margin="@dimen/fab_margin"
    android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>

```

Here is the result:



第103.4节：FloatingActionButton的行为设置

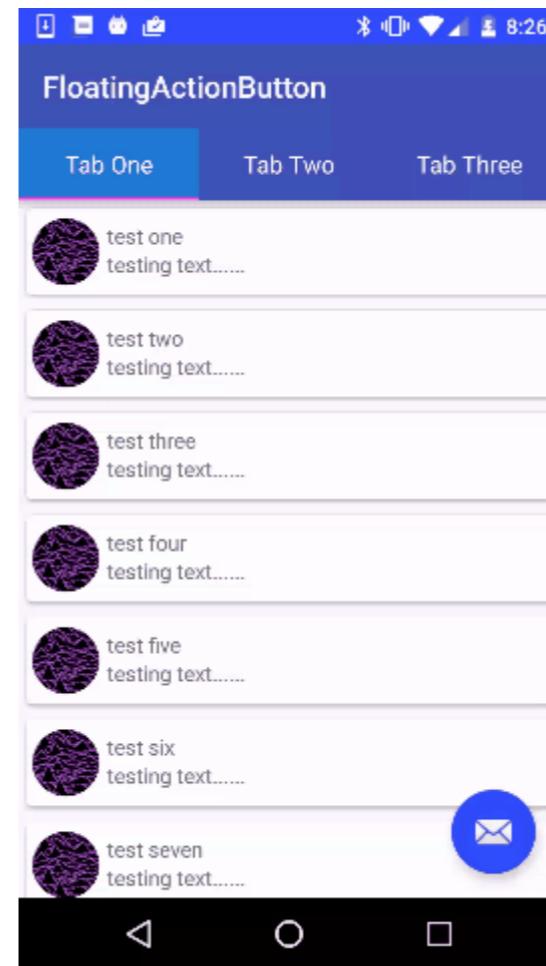
您可以在XML中设置FAB的行为。

例如：

```
<android.support.design.widget.FloatingActionButton  
    app:layout_behavior=".MyBehavior" />
```

或者您可以通过编程方式设置：

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();  
p.setBehavior(xxx);  
fab.setLayoutParams(p);
```



Section 103.4: Setting behaviour of FloatingActionButton

You can set the behavior of the FAB in XML.

For example:

```
<android.support.design.widget.FloatingActionButton  
    app:layout_behavior=".MyBehavior" />
```

Or you can set programmatically using:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();  
p.setBehavior(xxx);  
fab.setLayoutParams(p);
```

第104章：触摸事件

第104.1节：如何区分子视图和父视图组的触摸事件

- 嵌套视图组的onTouchEvents()可以通过boolean `onInterceptTouchEvent`来管理。

`OnInterceptTouchEvent`的默认值为false。

父视图的onTouchEvent会在子视图之前接收。如果`OnInterceptTouchEvent`返回false，则会将动作事件传递给子视图的`OnTouchEvent`处理器。如果返回true，则由父视图处理触摸事件。

但是，有时我们希望某些子元素管理`OnTouchEvent`，而某些则由父视图（或可能是父视图的父视图）管理。

这可以通过多种方式来管理。

- 保护子元素不被父视图的`OnInterceptTouchEvent`拦截的一种方法是实现`requestDisallowInterceptTouchEvent`。

```
public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)
```

如果该元素启用了事件处理器，这将阻止任何父视图管理该元素的`OnTouchEvent`。

如果`OnInterceptTouchEvent`为false，将评估子元素的`OnTouchEvent`。如果子元素中有处理各种触摸事件的方法，任何被禁用的相关事件处理器将把`OnTouchEvent`返回给父元素。

这个答案：

触摸事件传播通过的可视化示意：

父视图->子视图|父视图->子视图|父视图->子视图。

Chapter 104: Touch Events

Section 104.1: How to vary between child and parent view group touch events

- The `onTouchEvent()` for nested view groups can be managed by the boolean `onInterceptTouchEvent`.

The default value for the `OnInterceptTouchEvent` is false.

The parent's `onTouchEvent` is received before the child's. If the `OnInterceptTouchEvent` returns false, it sends the motion event down the chain to the child's `OnTouchEvent` handler. If it returns true the parent's will handle the touch event.

However there may be instances when we want some child elements to manage `OnTouchEvent`s and some to be managed by the parent view (or possibly the parent of the parent).

This can be managed in more than one way.

- One way a child element can be protected from the parent's `OnInterceptTouchEvent` is by implementing the `requestDisallowInterceptTouchEvent`.

```
public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)
```

This prevents any of the parent views from managing the `OnTouchEvent` for this element, if the element has event handlers enabled.

If the `OnInterceptTouchEvent` is false, the child element's `OnTouchEvent` will be evaluated. If you have a methods within the child elements handling the various touch events, any related event handlers that are disabled will return the `OnTouchEvent` to the parent.

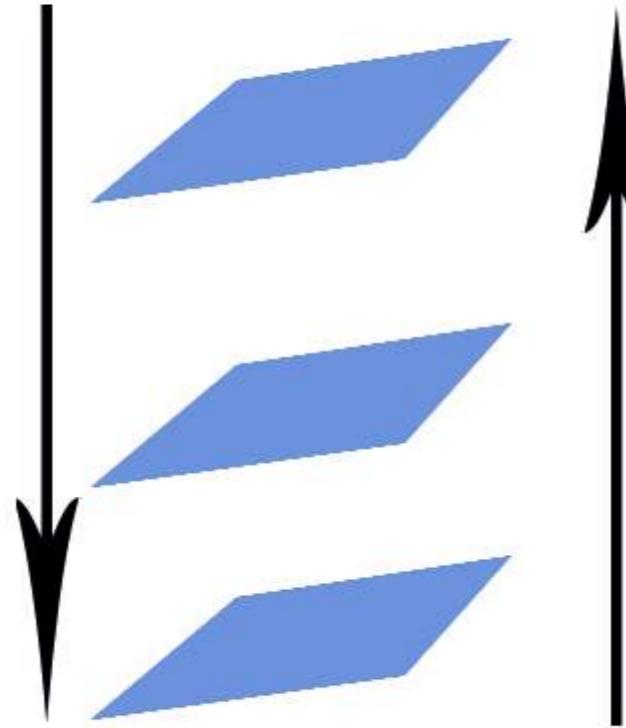
This answer:

A visualisation of how the propagation of touch events passes through:

parent -> child|parent -> child|parent -> child views.

Events will propagate until someone returns true!

onInterceptTouchEvent onTouchEvent



[来源于此处](#)

4. 另一种方法是从父视图的OnInterceptTouchEvent返回不同的值。

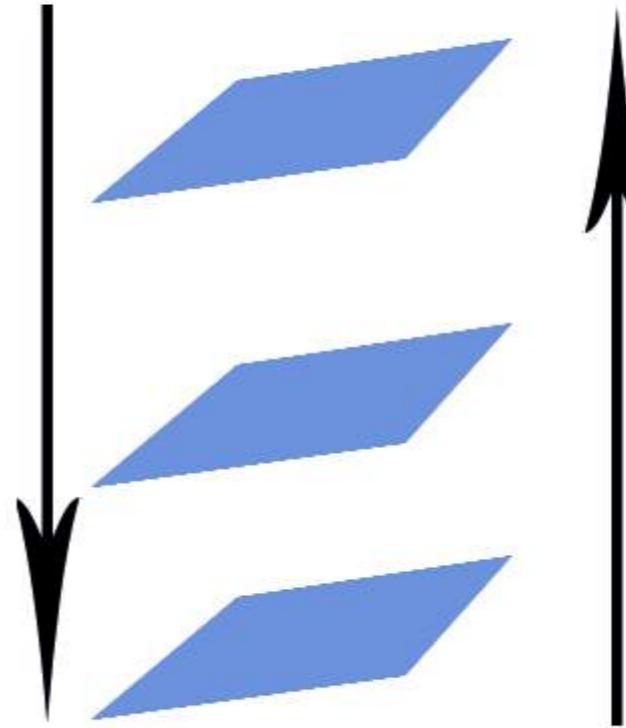
此示例取自Managing Touch Events in a ViewGroup，演示了如何在用户滚动时拦截子视图的OnTouchEvent。

4a.

```
@Override  
public boolean onInterceptTouchEvent(MotionEvent ev) {  
    /*  
     * 此方法仅用于确定我们是否想要拦截该动作。  
     * 如果返回true，将调用onTouchEvent并在那里执行实际的  
     * 滚动操作。  
     */  
  
    final int action = MotionEventCompat.getActionMasked(ev);  
  
    // 始终处理触摸手势完成的情况。  
    if (action == MotionEvent.ACTION_CANCEL || action == MotionEvent.ACTION_UP) {  
        // 释放滚动。  
        mIsScrolling = false;  
        return false; // 不拦截触摸事件，让子控件处理  
    }  
}
```

Events will propagate until someone returns true!

onInterceptTouchEvent onTouchEvent



[Courtesy from here](#)

4. Another way is returning varying values from the OnInterceptTouchEvent for the parent.

This example taken from [Managing Touch Events in a ViewGroup](#) and demonstrates how to intercept the child's OnTouchEvent when the user is scrolling.

4a.

```
@Override  
public boolean onInterceptTouchEvent(MotionEvent ev) {  
    /*  
     * This method JUST determines whether we want to intercept the motion.  
     * If we return true, onTouchEvent will be called and we do the actual  
     * scrolling there.  
     */  
  
    final int action = MotionEventCompat.getActionMasked(ev);  
  
    // Always handle the case of the touch gesture being complete.  
    if (action == MotionEvent.ACTION_CANCEL || action == MotionEvent.ACTION_UP) {  
        // Release the scroll.  
        mIsScrolling = false;  
        return false; // Do not intercept touch event, let the child handle it  
    }  
}
```

```

}

switch (action) {
    case MotionEvent.ACTION_MOVE: {
        if (mIsScrolling) {
            // 我们目前正在滚动，所以是的，拦截这个
            // 触摸事件！
            return true;
        }

        // 如果用户的手指水平拖动超过了
        // 触摸阈值，则开始滚动

        // 左移留给读者作为练习
        final int xDiff = calculateDistanceX(ev);

        // 触摸阈值应使用 ViewConfiguration
        // 常量计算。
        if (xDiff > mTouchSlop) {
            // 开始滚动！
        mIsScrolling = true;
        return true;
    }
    break;
}
...
}

// 通常情况下，我们不想拦截触摸事件。它们应该由子视图处理。

return false;
}

```

这是同一链接中的一些代码，展示如何创建元素周围矩形的参数：

4b.

```

// ImageButton 的点击区域
myButton.getHitRect(delegateArea);

// 将 ImageButton 的触摸区域扩展到其边界之外
// 右侧和底部。
delegateArea.right += 100;
delegateArea.bottom += 100;

// 实例化一个 TouchDelegate。
// "delegateArea" 是包含视图的本地坐标中的边界，
// 映射到代理视图。
// "myButton" 是应该接收触摸事件的子视图。

TouchDelegate touchDelegate = new TouchDelegate(delegateArea, myButton);

// 在父视图上设置 TouchDelegate，使得触摸
// 在触摸代理边界内的事件被路由到子视图。
if (View.class.isInstance(myButton.getParent())) {
    ((View) myButton.getParent()).setTouchDelegate(touchDelegate);
}

```

```

}

switch (action) {
    case MotionEvent.ACTION_MOVE: {
        if (mIsScrolling) {
            // We're currently scrolling, so yes, intercept the
            // touch event!
            return true;
        }

        // If the user has dragged her finger horizontally more than
        // the touch slop, start the scroll

        // left as an exercise for the reader
        final int xDiff = calculateDistanceX(ev);

        // Touch slop should be calculated using ViewConfiguration
        // constants.
        if (xDiff > mTouchSlop) {
            // Start scrolling!
            mIsScrolling = true;
            return true;
        }
    }
    break;
}
...
}

// In general, we don't want to intercept touch events. They should be
// handled by the child view.
return false;
}

```

This is some code from the same link showing how to create the parameters of the rectangle around your element:

4b.

```

// The hit rectangle for the ImageButton
myButton.getHitRect(delegateArea);

// Extend the touch area of the ImageButton beyond its bounds
// on the right and bottom.
delegateArea.right += 100;
delegateArea.bottom += 100;

// Instantiate a TouchDelegate.
// "delegateArea" is the bounds in local coordinates of
// the containing view to be mapped to the delegate view.
// "myButton" is the child view that should receive motion
// events.
TouchDelegate touchDelegate = new TouchDelegate(delegateArea, myButton);

// Sets the TouchDelegate on the parent view, such that touches
// within the touch delegate bounds are routed to the child.
if (View.class.isInstance(myButton.getParent())) {
    ((View) myButton.getParent()).setTouchDelegate(touchDelegate);
}

```

第105章：处理触摸和运动事件

监听器

onTouchListener 处理按钮、表面等的单次触摸

onTouchEvent 一种可以在表面（例如 SurfaceView）中找到的监听器。不需要像其他监听器（例如 onTouchListener）那样设置

onLongTouch 类似于 onTouch，但监听按钮、表面等的长按事件。

Android API 中一些基本触摸/运动处理系统的总结。

第105.1节：按钮

与按钮相关的触摸事件可以按如下方式检查：

```
public class ExampleClass extends Activity implements View.OnClickListener,
View.OnLongClickListener{
    public Button onLong, onClick;

    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        setContentView(R.layout.layout);
        onLong = (Button) findViewById(R.id.onLong);
        onClick = (Button) findViewById(R.id.onClick);
        // 按钮已创建。现在我们需要告诉系统
        // 这些按钮有监听器来检测触摸事件。
        // "this" 指的是这个类，因为它包含了相应的事件监听器。
        onLong.setOnLongClickListener(this);
        onClick.setOnClickListener(this);

        [或者]

        onClick.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                // 执行动作。此监听器仅设计用于一个按钮。
                // 这意味着这里不会有其他输入。
                // 这使得这里不需要使用 switch 语句。
            }
        });

        onLong.setOnLongClickListener(new View.OnLongClickListener(){
            @Override
            public boolean onLongClick(View v){
                // 参见 onClick.setOnClickListener() 中的注释。
            }
        });
    }

    @Override
    public void onClick(View v) {
        // 如果你需要处理多个按钮，使用 switch 来处理它们。
        switch(v.getId()){
            case R.id.onClick:
                // 执行动作。
                break;
        }
    }
}
```

详细信息

Chapter 105: Handling touch and motion events

Listener

onTouchListener Handles single touches for buttons, surfaces and more

onTouchEvent A listener that can be found in surfaces(e.g. SurfaceView). Does not need to be set like other listeners(e.g. onTouchListener)

onLongTouch Similar to onTouch, but listens for long presses in buttons, surfaces and more.

A summary of some of the basic touch/motion-handling systems in the Android API.

Section 105.1: Buttons

Touch events related to a [Button](#) can be checked as follows:

```
public class ExampleClass extends Activity implements View.OnClickListener,
View.OnLongClickListener{
    public Button onLong, onClick;

    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        setContentView(R.layout.layout);
        onLong = (Button) findViewById(R.id.onLong);
        onClick = (Button) findViewById(R.id.onClick);
        // The buttons are created. Now we need to tell the system that
        // these buttons have a listener to check for touch events.
        // "this" refers to this class, as it contains the appropriate event listeners.
        onLong.setOnLongClickListener(this);
        onClick.setOnClickListener(this);

        [OR]

        onClick.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                // Take action. This listener is only designed for one button.
                // This means, no other input will come here.
                // This makes a switch statement unnecessary here.
            }
        });

        onLong.setOnLongClickListener(new View.OnLongClickListener(){
            @Override
            public boolean onLongClick(View v){
                // See comment in onClick.setOnClickListener().
            }
        });
    }

    @Override
    public void onClick(View v) {
        // If you have several buttons to handle, use a switch to handle them.
        switch(v.getId()){
            case R.id.onClick:
                // Take action.
                break;
        }
    }
}
```

```

}

@Override
public boolean onLongClick(View v) {
    // 如果你需要处理多个按钮，使用 switch 来处理它们。
    switch(v.getId()){
        case R.id.onLong:
            // 执行动作。
            break;
    }
    return false;
}

```

第105.2节：表面

用于表面（例如 SurfaceView、GLSurfaceView 等）的触摸事件处理程序：

```

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;

public class ExampleClass extends Activity implements View.OnTouchListener{
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        CustomSurfaceView csv = new CustomSurfaceView(this);
        csv.setOnTouchListener(this);
        setContentView(csv);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // 如果处理多个视图，请添加 switch (参见按钮示例)
        // 这里你可以使用 MotionEvent event 来查看正在处理的触摸事件
        // 指针是按下还是抬起？是否在移动？
        return false;
    }
}

```

或者在表面中（替代方案）：

```

public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        super.onTouchEvent(ev);
        // 在这里处理触摸事件。这样做时，不需要调用监听器。
        // 请注意，此监听器仅适用于其所在的表面// (在本例中为 CustomSurfaceView)，这意味着在 SurfaceView 之外按下的任何其他内容// 都由应用程序中该区域具有监听器的部分处理。

        return true;
    }
}

```

```

}

@Override
public boolean onLongClick(View v) {
    // If you have several buttons to handle, use a switch to handle them.
    switch(v.getId()){
        case R.id.onLong:
            // Take action.
            break;
    }
    return false;
}

```

Section 105.2: Surface

Touch event handler for surfaces (e.g. SurfaceView, GLSurfaceView, and others):

```

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;

public class ExampleClass extends Activity implements View.OnTouchListener{
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        CustomSurfaceView csv = new CustomSurfaceView(this);
        csv.setOnTouchListener(this);
        setContentView(csv);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Add a switch (see buttons example) if you handle multiple views
        // here you can see (using MotionEvent event) to see what touch event
        // is being taken. Is the pointer touching or lifted? Is it moving?
        return false;
    }
}

```

Or alternatively (in the surface):

```

public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        super.onTouchEvent(ev);
        // Handle touch events here. When doing this, you do not need to call a listener.
        // Please note that this listener only applies to the surface it is placed in
        // (in this case, CustomSurfaceView), which means that anything else which is
        // pressed outside the SurfaceView is handled by the parts of your app that
        // have a listener in that area.

        return true;
    }
}

```

第105.3节：在表面上处理多点触控

```
public class CustomSurfaceView extends SurfaceView {  
    @Override  
    public boolean onTouchEvent(MotionEvent e) {  
        super.onTouchEvent(e);  
        if(e.getPointerCount() > 2){  
            return false; // 如果我们想限制指针数量，则返回 false  
            // 这将禁止该指针。对于任何未来的触摸事件，  
            // 直到该指针被抬起并重新按下，均不会对此作出反应。  
        }  
  
        // 这里你可以做什么？检查指针数量是否为 [x] 并采取相应操作，  
        // 如果有指针离开、新指针进入，或 [x] 个指针被移动。  
        // 一些处理触摸/移动事件的示例。  
  
        switch (MotionEventCompat.getActionMasked(e)) {  
            case MotionEvent.ACTION_DOWN:  
            case MotionEvent.ACTION_POINTER_DOWN:  
                // 一个或多个指针触摸屏幕。  
                break;  
            case MotionEvent.ACTION_UP:  
            case MotionEvent.ACTION_POINTER_UP:  
                // 一个或多个指针停止触摸屏幕。  
                break;  
            case MotionEvent.ACTION_MOVE:  
                // 一个或多个指针移动。  
                if(e.getPointerCount() == 2){  
                    move();  
                }else if(e.getPointerCount() == 1){  
                    paint();  
                }else{  
                    zoom();  
                }  
                break;  
        }  
        return true; // 允许重复动作。  
    }  
}
```

Section 105.3: Handling multitouch in a surface

```
public class CustomSurfaceView extends SurfaceView {  
    @Override  
    public boolean onTouchEvent(MotionEvent e) {  
        super.onTouchEvent(e);  
        if(e.getPointerCount() > 2){  
            return false; // If we want to limit the amount of pointers, we return false  
            // which disallows the pointer. It will not be reacted on either, for  
            // any future touch events until it has been lifted and repressed.  
        }  
  
        // What can you do here? Check if the amount of pointers are [x] and take action,  
        // if a pointer leaves, a new enters, or the [x] pointers are moved.  
        // Some examples as to handling etc. touch/motion events.  
  
        switch (MotionEventCompat.getActionMasked(e)) {  
            case MotionEvent.ACTION_DOWN:  
            case MotionEvent.ACTION_POINTER_DOWN:  
                // One or more pointers touch the screen.  
                break;  
            case MotionEvent.ACTION_UP:  
            case MotionEvent.ACTION_POINTER_UP:  
                // One or more pointers stop touching the screen.  
                break;  
            case MotionEvent.ACTION_MOVE:  
                // One or more pointers move.  
                if(e.getPointerCount() == 2){  
                    move();  
                }else if(e.getPointerCount() == 1){  
                    paint();  
                }else{  
                    zoom();  
                }  
                break;  
        }  
        return true; // Allow repeated action.  
    }  
}
```

第106章：在

Android中检测摇晃事件

第106.1节：Android示例中的摇晃检测器

```
public class ShakeDetector implements SensorEventListener {  
  
    private static final float SHAKE_THRESHOLD_GRAVITY = 2.7F;  
    private static final int SHAKE_SLOP_TIME_MS = 500;  
    private static final int SHAKE_COUNT_RESET_TIME_MS = 3000;  
  
    private OnShakeListener mListener;  
    private long mShakeTimestamp;  
    private int mShakeCount;  
  
    public void setOnShakeListener(OnShakeListener listener) {  
        this.mListener = listener;  
    }  
  
    public interface OnShakeListener {  
        public void onShake(int count);  
    }  
  
    @Override  
    public void onAccuracyChanged(Sensor sensor, int accuracy) {  
        // ignore  
    }  
  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
  
        if (mListener != null) {  
            float x = event.values[0];  
            float y = event.values[1];  
            float z = event.values[2];  
  
            float gX = x / SensorManager.GRAVITY_EARTH;  
            float gY = y / SensorManager.GRAVITY_EARTH;  
            float gZ = z / SensorManager.GRAVITY_EARTH;  
  
            // 当没有运动时, gForce 将接近 1。  
            float gForce = FloatMath.sqrt(gX * gX + gY * gY + gZ * gZ);  
  
            if (gForce > SHAKE_THRESHOLD_GRAVITY) {  
                final long now = System.currentTimeMillis();  
                // 忽略时间间隔过近的摇晃事件 (500毫秒)  
                if (mShakeTimestamp + SHAKE_SLOP_TIME_MS > now) {  
                    return;  
                }  
  
                // 在3秒无摇晃后重置摇晃计数  
                if (mShakeTimestamp + SHAKE_COUNT_RESET_TIME_MS < now) {  
                    mShakeCount = 0;  
                }  
  
                mShakeTimestamp = now;  
                mShakeCount++;  
            }  
        }  
    }  
}
```

Chapter 106: Detect Shake Event in Android

Section 106.1: Shake Detector in Android Example

```
public class ShakeDetector implements SensorEventListener {  
  
    private static final float SHAKE_THRESHOLD_GRAVITY = 2.7F;  
    private static final int SHAKE_SLOP_TIME_MS = 500;  
    private static final int SHAKE_COUNT_RESET_TIME_MS = 3000;  
  
    private OnShakeListener mListener;  
    private long mShakeTimestamp;  
    private int mShakeCount;  
  
    public void setOnShakeListener(OnShakeListener listener) {  
        this.mListener = listener;  
    }  
  
    public interface OnShakeListener {  
        public void onShake(int count);  
    }  
  
    @Override  
    public void onAccuracyChanged(Sensor sensor, int accuracy) {  
        // ignore  
    }  
  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
  
        if (mListener != null) {  
            float x = event.values[0];  
            float y = event.values[1];  
            float z = event.values[2];  
  
            float gX = x / SensorManager.GRAVITY_EARTH;  
            float gY = y / SensorManager.GRAVITY_EARTH;  
            float gZ = z / SensorManager.GRAVITY_EARTH;  
  
            // gForce will be close to 1 when there is no movement.  
            float gForce = FloatMath.sqrt(gX * gX + gY * gY + gZ * gZ);  
  
            if (gForce > SHAKE_THRESHOLD_GRAVITY) {  
                final long now = System.currentTimeMillis();  
                // ignore shake events too close to each other (500ms)  
                if (mShakeTimestamp + SHAKE_SLOP_TIME_MS > now) {  
                    return;  
                }  
  
                // reset the shake count after 3 seconds of no shakes  
                if (mShakeTimestamp + SHAKE_COUNT_RESET_TIME_MS < now) {  
                    mShakeCount = 0;  
                }  
  
                mShakeTimestamp = now;  
                mShakeCount++;  
            }  
        }  
    }  
}
```

```
mListener.onShake(mShakeCount);  
}  
}  
}
```

第106.2节：使用Seismic摇晃检测

Seismic 是Square开发的Android设备摇晃检测库。使用时只需开始监听它发出的摇晃事件。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);  
    sd = new ShakeDetector(() -> { /* 对检测到的摇晃做出反应 */ });  
}  
  
@Override  
protected void onResume() {  
    sd.start(sm);  
}  
  
@Override  
protected void onPause() {  
    sd.stop();  
}
```

要定义不同的加速度阈值，请使用 `sd.setSensitivity(sensitivity)`，其中 `sensitivity` 可以是 `SENSITIVITY_LIGHT`、`SENSITIVITY_MEDIUM`、`SENSITIVITY_HARD` 或任何其他合理的整数值。给定的默认值范围是 11 到 15。

安装

```
compile 'com.squareup:seismic:1.0.2'
```

```
mListener.onShake(mShakeCount);  
}  
}  
}
```

Section 106.2: Using Seismic shake detection

[Seismic](#) is an Android device shake detection library by Square. To use it just start listening to the shake events emitted by it.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);  
    sd = new ShakeDetector(() -> { /* react to detected shake */ });  
}  
  
@Override  
protected void onResume() {  
    sd.start(sm);  
}  
  
@Override  
protected void onPause() {  
    sd.stop();  
}
```

To define the a different acceleration threshold use `sd.setSensitivity(sensitivity)` with a sensitivity of `SENSITIVITY_LIGHT`, `SENSITIVITY_MEDIUM`, `SENSITIVITY_HARD` or any other reasonable integer value. The given default values range from 11 to 15.

Installation

```
compile 'com.squareup:seismic:1.0.2'
```

第107章：硬件按钮事件/意图 (PTT, LWP 等)

一些安卓设备由制造商添加了自定义按钮。这为开发者处理这些按钮打开了新的可能性，尤其是在为硬件设备制作应用时。

本主题记录了附带意图的按钮，您可以通过意图接收器监听这些意图。

第107.1节：Sonim设备

Sonim 设备根据型号有许多不同的自定义按键：

PTT_KEY

```
com.sonim.intent.action.PTT_KEY_DOWN  
com.sonim.intent.action.PTT_KEY_UP
```

YELLOW_KEY

```
com.sonim.intent.action.YELLOW_KEY_DOWN  
com.sonim.intent.action.YELLOW_KEY_UP
```

SOS_KEY

```
com.sonim.intent.action.SOS_KEY_DOWN  
com.sonim.intent.action.SOS_KEY_UP
```

GREEN_KEY

```
com.sonim.intent.action.GREEN_KEY_DOWN  
com.sonim.intent.action.GREEN_KEY_UP
```

注册按键

要接收这些意图，您需要在手机设置中将按钮分配给您的应用。Sonim有一个功能，可以在应用安装时自动将按钮注册到应用。为此，您需要联系他们并获取一个特定于包的密钥，包含在您的Manifest中，如下所示：

```
<meta-data  
    android:name="app_key_green_data"  
    android:value="your-key-here" />
```

第107.2节：RugGear设备

PTT按钮

```
android.intent.action.PTT.down  
android.intent.action.PTT.up
```

已确认设备：RG730, RG740A

Chapter 107: Hardware Button Events/Intents (PTT, LWP, etc.)

Several android devices have custom buttons added by the manufacturer. This opens new possibilities for the developer in handling those buttons especially when making Apps targeted for Hardware Devices.

This topic documents buttons which have intents attached to them which you can listen for via intent-receivers.

Section 107.1: Sonim Devices

Sonim devices have varying by model a lot of different custom buttons:

PTT_KEY

```
com.sonim.intent.action.PTT_KEY_DOWN  
com.sonim.intent.action.PTT_KEY_UP
```

YELLOW_KEY

```
com.sonim.intent.action.YELLOW_KEY_DOWN  
com.sonim.intent.action.YELLOW_KEY_UP
```

SOS_KEY

```
com.sonim.intent.action.SOS_KEY_DOWN  
com.sonim.intent.action.SOS_KEY_UP
```

GREEN_KEY

```
com.sonim.intent.action.GREEN_KEY_DOWN  
com.sonim.intent.action.GREEN_KEY_UP
```

Registering the buttons

To receive those intents you will have to assign the buttons to your app in the Phone-Settings. Sonim has a possibility to auto-register the buttons to the App when it is installed. In order to do that you will have to contact them and get a package-specific key to include in your Manifest like this:

```
<meta-data  
    android:name="app_key_green_data"  
    android:value="your-key-here" />
```

Section 107.2: RugGear Devices

PTT Button

```
android.intent.action.PTT.down  
android.intent.action.PTT.up
```

Confirmed on: RG730, RG740A

第108章：GreenRobot EventBus

线程模式	描述
ThreadMode.POSTING	将在事件发布的同一线程上调用。这是默认模式。
ThreadMode主线程	将在主UI线程上调用。
ThreadMode后台线程	将在后台线程上调用。如果发布线程不是主线程，则会使用该线程。如果在主线程上发布，EventBus有一个单独的后台线程将被使用。
ThreadMode异步线程	将在其自己的线程上调用。

第108.1节：传递一个简单事件

我们首先需要做的是将 EventBus 添加到模块的 gradle 文件中：

```
dependencies {  
    ...  
    compile 'org.greenrobot:eventbus:3.0.0'  
    ...  
}
```

现在我们需要为我们的事件创建一个模型。它可以包含我们想要传递的任何内容。现在我们先创建一个空类。

```
public class DeviceConnectedEvent  
{  
}
```

现在我们可以向我们的Activity添加代码，注册EventBus并订阅该事件。

```
public class MainActivity extends AppCompatActivity  
{  
    private EventBus _eventBus;  
  
    @Override  
    protected void onCreate (Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        _eventBus = EventBus.getDefault();  
    }  
  
    @Override  
    protected void onStart ()  
    {  
        super.onStart();  
        _eventBus.register(this);  
    }  
  
    @Override  
    protected void onStop ()  
    {  
        _eventBus.unregister(this);  
        super.onStop();  
    }  
  
    @Subscribe(threadMode = ThreadMode.MAIN)  
}
```

Chapter 108: GreenRobot EventBus

Thread Mode	Description
ThreadMode.POSTING	Will be called on the same thread that the event was posted on. This is the default mode.
ThreadMode.MAIN	Will be called on the main UI thread.
ThreadMode.BACKGROUND	Will be called on a background thread. If the posting thread isn't the main thread it will be used. If posted on the main thread EventBus has a single background thread that it will use.
ThreadMode.ASYNC	Will be called on its own thread.

Section 108.1: Passing a Simple Event

The first thing we need to do is add EventBus to our module's gradle file:

```
dependencies {  
    ...  
    compile 'org.greenrobot:eventbus:3.0.0'  
    ...  
}
```

Now we need to create a model for our event. It can contain anything we want to pass along. For now we'll just make an empty class.

```
public class DeviceConnectedEvent  
{  
}
```

Now we can add the code to our Activity that will register with EventBus and subscribe to the event.

```
public class MainActivity extends AppCompatActivity  
{  
    private EventBus _eventBus;  
  
    @Override  
    protected void onCreate (Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        _eventBus = EventBus.getDefault();  
    }  
  
    @Override  
    protected void onStart ()  
    {  
        super.onStart();  
        _eventBus.register(this);  
    }  
  
    @Override  
    protected void onStop ()  
    {  
        _eventBus.unregister(this);  
        super.onStop();  
    }  
  
    @Subscribe(threadMode = ThreadMode.MAIN)  
}
```

```

public void onDeviceConnected (final DeviceConnectedEvent event)
{
    // 处理事件并更新界面
}

```

在这个Activity中，我们在onCreate()方法中获取了EventBus的实例。我们在onStart()和onStop()中注册/注销事件。重要的是要记得在监听器失去作用域时注销，否则可能会导致Activity泄漏。

最后我们定义了希望被事件调用的方法。@Subscribe注解告诉EventBus可以查找哪些方法来处理事件。你必须至少有一个用@Subscribe注解的方法才能注册到EventBus，否则会抛出异常。在注解中我们定义了线程模式。这告诉EventBus在哪个线程调用该方法。这是一个非常方便的方式，将信息从后台线程传递到UI线程！这正是我们这里所做的。ThreadMode.MAIN表示该方法将在Android的主UI线程上调用，因此可以安全地进行任何需要的UI操作。方法名无关紧要。除了@Subscribe注解外，EventBus唯一关注的是参数类型。只要类型匹配，事件发布时该方法就会被调用。

最后我们需要做的是发布一个事件。这段代码将在我们的Service中。

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

就这么简单！EventBus会接收这个DeviceConnectedEvent，遍历其注册的监听器，查找它们订阅的方法，找到那些以DeviceConnectedEvent作为参数的方法，并在它们指定的线程上调用这些方法。

第108.2节：接收事件

要接收事件，您需要在EventBus上注册您的类。

```

@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
    EventBus.getDefault().unregister(this);
    super.onStop();
}

```

然后订阅事件。

```

@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitraryEvent event) {
    Toast.makeText(getApplicationContext(), "事件类型: "+event.getEventType(), Toast.LENGTH_SHORT).show();
}

```

第108.3节：发送事件

发送事件就像创建事件对象然后发布它一样简单。

```
EventBus.getDefault().post(new ArbitraryEvent(ArbitraryEvent.TYPE_1));
```

```

public void onDeviceConnected (final DeviceConnectedEvent event)
{
    // Process event and update UI
}

```

In this Activity we get an instance of EventBus in the onCreate() method. We register / unregister for events in onStart() / onStop(). It's important to remember to unregister when your listener loses scope or you could leak your Activity.

Finally we define the method that we want called with the event. The @Subscribe annotation tells EventBus which methods it can look for to handle events. You have to have at least one methods annotated with @Subscribe to register with EventBus or it will throw an exception. In the annotation we define the thread mode. This tells EventBus which thread to call the method on. It is a very handy way of passing information from a background thread to the UI thread! That's exactly what we're doing here. ThreadMode.MAIN means that this method will be called on Android's main UI thread so it's safe to do any UI manipulations here that you need. The name of the method doesn't matter. The only think, other than the @Subscribe annotation, that EventBus is looking for is the type of the argument. As long as the type matches it will be called when an event is posted.

The last thing we need to do is to post an event. This code will be in our Service.

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

That's all there is to it! EventBus will take that DeviceConnectedEvent and look through its registered listeners, look through the methods that they've subscribed and find the ones that take a DeviceConnectedEvent as an argument and call them on the thread that they want to be called on.

Section 108.2: Receiving Events

For receiving events you need to register your class on the EventBus.

```

@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
    EventBus.getDefault().unregister(this);
    super.onStop();
}

```

And then subscribe to the events.

```

@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitraryEvent event) {
    Toast.makeText(getApplicationContext(), "Event type: "+event.getEventType(), Toast.LENGTH_SHORT).show();
}

```

Section 108.3: Sending Events

Sending events is as easy as creating the Event object and then posting it.

```
EventBus.getDefault().post(new ArbitraryEvent(ArbitraryEvent.TYPE_1));
```

第109章：Otto事件总线

第109.1节：传递事件

本示例描述了如何使用Otto事件总线传递事件。

要在Android Studio中使用Otto事件总线，必须在模块的gradle文件中插入以下语句：

```
dependencies {  
    compile 'com.squareup:otto:1.3.8'  
}
```

我们想要传递的事件是一个简单的Java对象：

```
public class DatabaseContentChangedEvent {  
    public String message;  
  
    public DatabaseContentChangedEvent(String message) {  
        this.message = message;  
    }  
}
```

我们需要一个Bus来发送事件。通常这是一个单例：

```
import com.squareup.otto.Bus;  
  
public final class BusProvider {  
    private static final Bus mBus = new Bus();  
  
    public static Bus getInstance() {  
        return mBus;  
    }  
  
    private BusProvider() {  
    }  
}
```

发送事件时，我们只需要使用 BusProvider 及其 post 方法。这里当一个 AsyncTask 的操作完成时发送一个事件：

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {  
  
    ...  
  
    @Override  
    protected void onPostExecute(Void param) {  
        BusProvider.getInstance().post(  
            new DatabaseContentChangedEvent("内容已更改")  
        );  
    }  
}
```

第109.2节：接收事件

要接收事件，必须实现一个以事件类型为参数的方法，并使用注解 @Subscribe。此外，您还必须在 BusProvider 注册/注销您的对象实例（参见

Chapter 109: Otto Event Bus

Section 109.1: Passing an event

This example describes passing an event using the [Otto Event Bus](#).

To use the Otto Event Bus in **Android Studio** you have to insert the following statement in your modules gradle file:

```
dependencies {  
    compile 'com.squareup:otto:1.3.8'  
}
```

The event we'd like to pass is a simple Java object:

```
public class DatabaseContentChangedEvent {  
    public String message;  
  
    public DatabaseContentChangedEvent(String message) {  
        this.message = message;  
    }  
}
```

We need a Bus to send events. This is typically a singleton:

```
import com.squareup.otto.Bus;  
  
public final class BusProvider {  
    private static final Bus mBus = new Bus();  
  
    public static Bus getInstance() {  
        return mBus;  
    }  
  
    private BusProvider() {  
    }  
}
```

To send an event we only need our BusProvider and its post method. Here we send an event if the action of an AsyncTask is completed:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {  
  
    ...  
  
    @Override  
    protected void onPostExecute(Void param) {  
        BusProvider.getInstance().post(  
            new DatabaseContentChangedEvent("Content changed")  
        );  
    }  
}
```

Section 109.2: Receiving an event

To receive an event it is necessary to implement a method with the event type as parameter and annotate it using @Subscribe. Furthermore you have to register/unregister the instance of your object at the BusProvider (see

示例 发送事件):

```
public class MyFragment extends Fragment {  
    private final static String TAG = "MyFragment";  
  
    ...  
  
    @Override  
    public void onResume() {  
        super.onResume();  
        BusProvider.getInstance().register(this);  
    }  
  
    @Override  
    public void onPause() {  
        super.onPause();  
        BusProvider.getInstance().unregister(this);  
    }  
  
    @Subscribe  
    public void onDatabaseContentChanged(DatabaseContentChangedEvent event) {  
        Log.i(TAG, "onDatabaseContentChanged: "+event.message);  
    }  
}
```

重要提示：为了接收该事件，类的实例必须存在。通常在你想从一个活动向另一个活动发送结果时，情况并非如此。因此请检查你对事件总线的使用场景。

example *Sending an event*):

```
public class MyFragment extends Fragment {  
    private final static String TAG = "MyFragment";  
  
    ...  
  
    @Override  
    public void onResume() {  
        super.onResume();  
        BusProvider.getInstance().register(this);  
    }  
  
    @Override  
    public void onPause() {  
        super.onPause();  
        BusProvider.getInstance().unregister(this);  
    }  
  
    @Subscribe  
    public void onDatabaseContentChanged(DatabaseContentChangedEvent event) {  
        Log.i(TAG, "onDatabaseContentChanged: "+event.message);  
    }  
}
```

Important: In order to receive that event an instance of the class has to exist. This is usually not the case when you want to send a result from one activity to another activity. So check your use case for the event bus.

第110章：振动

第110.1节：振动入门

授予振动权限

在开始实现代码之前，您必须在安卓清单文件中添加权限：

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

导入振动库

```
import android.os.Vibrator;
```

从上下文中获取 Vibrator 实例

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

检查设备是否有振动器

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

第110.2节：无限振动

使用 `vibrate(long[] pattern, int repeat)`

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

```
// 启动时间延迟
// 振动500毫秒
// 休眠1000毫秒
long[] pattern = {0, 500, 1000};

// 0 表示无限重复
vibrator.vibrate(pattern, 0);
```

第110.3节：振动模式

您可以通过传入一个long类型数组来创建振动模式，数组中的每个元素表示持续时间，单位为毫秒。第一个数字是开始时间延迟。数组中的每个元素依次交替表示振动、暂停、振动、暂停，依此类推。

以下示例演示了该模式：

- 振动100毫秒，暂停1000毫秒
- 振动200毫秒，暂停2000毫秒

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Chapter 110: Vibration

Section 110.1: Getting Started with Vibration

Grant Vibration Permission

before you start implement code, you have to add permission in android manifest :

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Import Vibration Library

```
import android.os.Vibrator;
```

Get instance of Vibrator from Context

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Check device has vibrator

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Section 110.2: Vibrate Indefinitely

using the `vibrate(long[] pattern, int repeat)`

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

// Start time delay
// Vibrate for 500 milliseconds
// Sleep for 1000 milliseconds
long[] pattern = {0, 500, 1000};

// 0 meaning is repeat indefinitely
vibrator.vibrate(pattern, 0);
```

Section 110.3: Vibration Patterns

You can create vibration patterns by passing in an array of longs, each of which represents a duration in milliseconds. The first number is start time delay. Each array entry then alternates between vibrate, sleep, vibrate, sleep, etc.

The following example demonstrates this pattern:

- vibrate 100 milliseconds and sleep 1000 milliseconds
- vibrate 200 milliseconds and sleep 2000 milliseconds

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

要使模式重复，传入模式数组中开始重复的索引，传入-1则禁用重复。

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // 不重复
vibrator.vibrate(pattern, 0); // 无限重复
```

第110.4节：停止振动

如果您想停止振动，请调用：

```
vibrator.cancel();
```

第110.5节：振动一次

使用*vibrate(long milliseconds)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

To cause the pattern to repeat, pass in the index into the pattern array at which to start the repeat, or -1 to disable repeating.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern, 0); // repeats forever
```

Section 110.4: Stop Vibrate

If you want stop vibrate please call :

```
vibrator.cancel();
```

Section 110.5: Vibrate for one time

using the *vibrate(long milliseconds)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

第111章：内容提供者 (ContentProvider)

第111.1节：实现一个基本的内容提供者类

1) 创建一个契约类 (Contract Class)

合同类定义了帮助应用程序使用内容URI、列名、意图操作以及内容提供者的其他功能的常量。合同类不会自动包含在提供者中；提供者的开发者必须定义它们，然后将其提供给其他开发者使用。

提供者通常只有一个权限，作为其 Android 内部名称。为了避免与其他提供者冲突，请使用唯一的权限。由于这一建议对 Android 包名同样适用，您可以将提供者权限定义为包含该提供者的包名的扩展。例如，如果您的 Android 包名是 com.example.app name，您应该将提供者权限命名为 com.example.appname.provider。

```
public class MyContract {  
    public static final String CONTENT_AUTHORITY = "com.example.myApp";  
    public static final String PATH_DATABASE = "dataTable";  
    public static final String TABLE_NAME = "dataTable";  
}
```

内容 URI 是用于标识提供者中数据的 URI。内容 URI 包含整个提供者的符号名称（其权限）和指向表或文件的名称（路径）。可选的 id 部分指向表中的单个行。ContentProvider 的每个数据访问方法都以内容 URI 作为参数；这使您能够确定要访问的表、行或文件。在契约类中定义这些内容。

```
public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);  
public static final Uri CONTENT_URI =  
BASE_CONTENT_URI.buildUpon().appendPath(PATH_DATABASE).build();  
  
// 定义表的所有列和所需的通用函数
```

2) 创建辅助类

辅助类负责管理数据库的创建和版本管理。

```
public class DatabaseHelper extends SQLiteOpenHelper {  
  
    // 当数据库结构发生变化时，增加版本号  
    public static final int DATABASE_VERSION = 1;  
    // 文件系统中数据库的名称，可以自行选择  
    public static final String DATABASE_NAME = "weather.db";  
  
    public DatabaseHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        // 当数据库首次创建时调用。在这里应创建表并初始化表数据。  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

Chapter 111: ContentProvider

Section 111.1: Implementing a basic content provider class

1) Create a Contract Class

A contract class defines constants that help applications work with the content URIs, column names, intent actions, and other features of a content provider. Contract classes are not included automatically with a provider; the provider's developer has to define them and then make them available to other developers.

A provider usually has a single authority, which serves as its Android-internal name. To avoid conflicts with other providers, use a unique content authority. Because this recommendation is also true for Android package names, you can define your provider authority as an extension of the name of the package containing the provider. For example, if your Android package name is com.example.appname, you should give your provider the authority com.example.appname.provider.

```
public class MyContract {  
    public static final String CONTENT_AUTHORITY = "com.example.myApp";  
    public static final String PATH_DATABASE = "dataTable";  
    public static final String TABLE_NAME = "dataTable";  
}
```

A content URI is a URI that identifies data in a provider. Content URIs include the symbolic name of the entire provider (its authority) and a name that points to a table or file (a path). The optional id part points to an individual row in a table. Every data access method of ContentProvider has a content URI as an argument; this allows you to determine the table, row, or file to access. Define these in the contract class.

```
public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);  
public static final Uri CONTENT_URI =  
BASE_CONTENT_URI.buildUpon().appendPath(PATH_DATABASE).build();  
  
// define all columns of table and common functions required
```

2) Create the Helper Class

A helper class manages database creation and version management.

```
public class DatabaseHelper extends SQLiteOpenHelper {  
  
    // Increment the version when there is a change in the structure of database  
    public static final int DATABASE_VERSION = 1;  
    // The name of the database in the filesystem, you can choose this to be anything  
    public static final String DATABASE_NAME = "weather.db";  
  
    public DatabaseHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        // Called when the database is created for the first time. This is where the  
        // creation of tables and the initial population of the tables should happen.  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
// 当数据库需要升级时调用。实现中应使用此方法删除表、添加表，或执行其他升级到新架构版本所需的操作。
```

```
}
```

3) 创建一个继承自 ContentProvider 类的类

```
public class MyProvider extends ContentProvider {  
  
    public DatabaseHelper dbHelper;  
  
    public static final UriMatcher matcher = buildUriMatcher();  
    public static final int DATA_TABLE = 100;  
    public static final int DATA_TABLE_DATE = 101;
```

UriMatcher 将一个权限 (authority) 和路径映射到一个整数值。方法 match() 返回一个 URI 的唯一整数值 (可以是任意数字，只要唯一即可)。switch 语句用于选择查询整个表，还是查询单条记录。我们的 UriMatcher 如果 URI 是表的内容 URI，则返回 100；如果 URI 指向该表中的特定行，则返回 101。你可以使用 # 通配符匹配任意数字，使用 * 匹配任意字符串。

```
public static UriMatcher buildUriMatcher() {  
    UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);  
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE, DATA_TABLE);  
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE + "/#", DATA_TABLE_DATE);  
    return uriMatcher;  
}
```

重要提示： addURI() 调用的顺序很重要！UriMatcher 会按添加顺序依次查找。由于通配符如 # 和 * 是贪婪匹配，你需要确保 URI 的顺序正确。例如：

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);  
uriMatcher.addURI(CONTENT_AUTHORITY, "/*", 2);
```

这是正确的顺序，因为匹配器会先查找 /example，然后才会匹配 /*。如果这两个调用顺序颠倒，且你调用 uriMatcher.match("/example")，那么 UriMatcher 会在遇到 /* 路径时停止查找并返回错误结果！

你需要重写以下函数：

onCreate(): 初始化你的提供者。Android 系统在创建你的提供者后立即调用此方法。注意，只有当 ContentResolver 对象尝试访问时，你的提供者才会被创建。

```
@Override  
public boolean onCreate() {  
    dbhelper = new DatabaseHelper(getContext());  
    return true;  
}
```

getType(): 返回对应内容URI的MIME类型

```
@Override  
public String getType(Uri uri) {  
    final int match = matcher.match(uri);
```

```
// Called when the database needs to be upgraded. The implementation  
// should use this method to drop tables, add tables, or do anything else it  
// needs to upgrade to the new schema version.
```

```
}
```

3) Create a class that extends ContentProvider class

```
public class MyProvider extends ContentProvider {  
  
    public DatabaseHelper dbHelper;  
  
    public static final UriMatcher matcher = buildUriMatcher();  
    public static final int DATA_TABLE = 100;  
    public static final int DATA_TABLE_DATE = 101;
```

A UriMatcher maps an authority and path to an integer value. The method match() returns a unique integer value for a URI (it can be any arbitrary number, as long as it's unique). A switch statement chooses between querying the entire table, and querying for a single record. Our UriMatcher returns 100 if the URI is the Content URI of Table and 101 if the URI points to a specific row within that table. You can use the # wildcard to match with any number and * to match with any string.

```
public static UriMatcher buildUriMatcher() {  
    UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);  
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE, DATA_TABLE);  
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE + "/#", DATA_TABLE_DATE);  
    return uriMatcher;  
}
```

IMPORTANT: the ordering of addURI() calls matters! The UriMatcher will look in sequential order from first added to last. Since wildcards like # and * are greedy, you will need to make sure that you have ordered your URIs correctly. For example:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);  
uriMatcher.addURI(CONTENT_AUTHORITY, "/*", 2);
```

is the proper ordering, since the matcher will look for /example first before resorting to the /* match. If these method calls were reversed and you called uriMatcher.match("/example")，then the UriMatcher will stop looking for matches once it encounters the /* path and return the wrong result!

You will then need to override these functions:

onCreate(): Initialize your provider. The Android system calls this method immediately after it creates your provider. Notice that your provider is not created until a ContentResolver object tries to access it.

```
@Override  
public boolean onCreate() {  
    dbhelper = new DatabaseHelper(getContext());  
    return true;  
}
```

getType(): Return the MIME type corresponding to a content URI

```
@Override  
public String getType(Uri uri) {  
    final int match = matcher.match(uri);
```

```

switch (match) {
    case DATA_TABLE:
        return ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATABASE;
    case DATA_TABLE_DATE:
        return ContentResolver.ANY_CURSOR_ITEM_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATABASE;
    default:
        throw new UnsupportedOperationException("Unknown Uri: " + uri);
}

```

query(): 从您的提供者检索数据。使用参数选择要查询的表、要返回的行和列，以及结果的排序顺序。以 Cursor 对象的形式返回数据。

```

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}

```

向您的提供者插入一行新数据。使用参数选择目标表并获取要使用的列值。返回新插入行的内容URI。

```

@Override
public Uri insert(Uri uri, ContentValues values)
{
    final SQLiteDatabase db = dbHelper.getWritableDatabase();
    long id = db.insert(MyContract.TABLE_NAME, null, values);
    return ContentUris.withAppendedId(MyContract.CONTENT_URI, id);
}

```

delete(): 从您的提供者中删除行。使用参数选择要删除的表和行。返回删除的行数。

```

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsDeleted = db.delete(MyContract.TABLE_NAME, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsDeleted;
}

```

update(): 更新提供者中的现有行。使用参数选择要更新的表和行，并获取新的列值。返回更新的行数。

```

@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsUpdated = db.update(MyContract.TABLE_NAME, values, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsUpdated;
}

```

```

switch (match) {
    case DATA_TABLE:
        return ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATABASE;
    case DATA_TABLE_DATE:
        return ContentResolver.ANY_CURSOR_ITEM_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATABASE;
    default:
        throw new UnsupportedOperationException("Unknown Uri: " + uri);
}

```

query(): Retrieve data from your provider. Use the arguments to select the table to query, the rows and columns to return, and the sort order of the result. Return the data as a Cursor object.

```

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}

```

Insert a new row into your provider. Use the arguments to select the destination table and to get the column values to use. Return a content URI for the newly-inserted row.

```

@Override
public Uri insert(Uri uri, ContentValues values)
{
    final SQLiteDatabase db = dbHelper.getWritableDatabase();
    long id = db.insert(MyContract.TABLE_NAME, null, values);
    return ContentUris.withAppendedId(MyContract.CONTENT_URI, id);
}

```

delete(): Delete rows from your provider. Use the arguments to select the table and the rows to delete. Return the number of rows deleted.

```

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsDeleted = db.delete(MyContract.TABLE_NAME, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsDeleted;
}

```

update(): Update existing rows in your provider. Use the arguments to select the table and rows to update and to get the new column values. Return the number of rows updated.

```

@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsUpdated = db.update(MyContract.TABLE_NAME, values, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsUpdated;
}

```

4) 更新清单文件

```
<provider  
    android:authorities="com.example.myApp"  
    android:name=".DatabaseProvider" />
```

4) Update manifest file

```
<provider  
    android:authorities="com.example.myApp"  
    android:name=".DatabaseProvider" />
```

第112章：Dagger 2

第112.1节：应用程序和活动注入的组件设置

一个基本的AppComponent，依赖于单个 AppModule 来提供应用范围内的单例对象。

```
@Singleton  
@Component(modules = AppModule.class)  
public interface AppComponent {  
  
    void inject(App app);  
  
    Context provideContext();  
  
    Gson provideGson();  
}
```

一个与AppComponent一起使用的模块，将提供其单例对象，例如整个应用中重用的Gson实例。

```
@Module  
public class AppModule {  
  
    private final Application mApplication;  
  
    public AppModule(Application application) {  
        mApplication = application;  
    }  
  
    @Singleton  
    @Provides  
    Gson provideGson() {  
        return new Gson();  
    }  
  
    @Singleton  
    @Provides  
    Context provideContext() {  
        return mApplication;  
    }  
}
```

一个用于设置Dagger和单例组件的子类化应用程序。

```
public class App extends Application {  
  
    @Inject  
    AppComponent mAppComponent;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
  
        DaggerAppComponent.builder().appModule(new AppModule(this)).build().inject(this);  
    }  
  
    public AppComponent getAppComponent() {
```

Chapter 112: Dagger 2

Section 112.1: Component setup for Application and Activity injection

A basic AppComponent that depends on a single AppModule to provide application-wide singleton objects.

```
@Singleton  
@Component(modules = AppModule.class)  
public interface AppComponent {  
  
    void inject(App app);  
  
    Context provideContext();  
  
    Gson provideGson();  
}
```

A module to use together with the AppComponent which will provide its singleton objects, e.g. an instance of Gson to reuse throughout the whole application.

```
@Module  
public class AppModule {  
  
    private final Application mApplication;  
  
    public AppModule(Application application) {  
        mApplication = application;  
    }  
  
    @Singleton  
    @Provides  
    Gson provideGson() {  
        return new Gson();  
    }  
  
    @Singleton  
    @Provides  
    Context provideContext() {  
        return mApplication;  
    }  
}
```

A subclassed application to setup dagger and the singleton component.

```
public class App extends Application {  
  
    @Inject  
    AppComponent mAppComponent;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
  
        DaggerAppComponent.builder().appModule(new AppModule(this)).build().inject(this);  
    }  
  
    public AppComponent getAppComponent() {
```

```

    return mAppComponent;
}
}

```

现在是一个依赖于AppComponent以访问单例对象的活动作用域组件。

```

@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

```

以及一个可重用的ActivityModule，用于提供基本依赖，如FragmentManager

```

@Module
public class ActivityModule {

    private final AppCompatActivity mActivity;

    public ActivityModule(AppCompatActivity activity) {
        mActivity = activity;
    }

    @ActivityScope
    public AppCompatActivity provideActivity() {
        return mActivity;
    }

    @ActivityScope
    public FragmentManager 提供FragmentManager(AppCompatActivity 活动) {
        return 活动.getSupportFragmentManager();
    }
}

```

把所有东西放在一起，我们已经设置好，可以注入我们的活动，并确保在整个应用中使用相同的Gson！

```

public class MainActivity 继承 AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DaggerMainActivityComponent.builder()
            .appComponent(((App) getApplication()).getAppComponent())
            .activityModule(new ActivityModule(this))
        .build().inject(this);
    }
}

```

第112.2节：自定义作用域

@Scope

```

    return mAppComponent;
}
}

```

Now an activity scoped component that depends on the AppComponent to gain access to the singleton objects.

```

@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

```

And a reusable ActivityModule that will provide basic dependencies, like a FragmentManager

```

@Module
public class ActivityModule {

    private final AppCompatActivity mActivity;

    public ActivityModule(AppCompatActivity activity) {
        mActivity = activity;
    }

    @ActivityScope
    public AppCompatActivity provideActivity() {
        return mActivity;
    }

    @ActivityScope
    public FragmentManager provideFragmentManager(AppCompatActivity activity) {
        return activity.getSupportFragmentManager();
    }
}

```

Putting everything together we're set up and can inject our activity and be sure to use the same Gson throughout our app!

```

public class MainActivity extends AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DaggerMainActivityComponent.builder()
            .appComponent(((App) getApplication()).getAppComponent())
            .activityModule(new ActivityModule(this))
        .build().inject(this);
    }
}

```

Section 112.2: Custom Scopes

@Scope

```
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

作用域只是注解，您可以根据需要创建自己的注解。

第112.3节：使用 @Subcomponent 替代 @Component(dependencies={...})

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {
    void inject(App app);

    Context provideContext();
    Gson provideGson();

    MainActivityComponent mainActivityComponent(ActivityModule activityModule);
}

@ActivityScope
@Subcomponent(modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

public class MainActivity 继承 AppCompatActivity {
    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ((App) getApplication()).getAppComponent()
            .mainActivityComponent(new ActivityModule(this)).inject(this);
    }
}
```

第112.4节：从多个模块创建组件

Dagger 2 支持从多个模块创建组件。你可以这样创建你的组件：

```
@Singleton
@Component(modules = {GeneralPurposeModule.class, SpecificModule.class})
public interface MyMultipleModuleComponent {
    void inject(MyFragment myFragment);
    void inject(MyService myService);
    void inject(MyController myController);
    void inject(MyActivity myActivity);
}
```

这两个引用模块 GeneralPurposeModule 和 SpecificModule 可以按如下方式实现：

GeneralPurposeModule.java

```
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Scopes are just annotations and you can create your own ones where needed.

Section 112.3: Using @Subcomponent instead of @Component(dependencies={...})

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {
    void inject(App app);

    Context provideContext();
    Gson provideGson();

    MainActivityComponent mainActivityComponent(ActivityModule activityModule);
}

@ActivityScope
@Subcomponent(modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

public class MainActivity extends AppCompatActivity {
    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ((App) getApplication()).getAppComponent()
            .mainActivityComponent(new ActivityModule(this)).inject(this);
    }
}
```

Section 112.4: Creating a component from multiple modules

Dagger 2 supports creating a component from multiple modules. You can create your component this way:

```
@Singleton
@Component(modules = {GeneralPurposeModule.class, SpecificModule.class})
public interface MyMultipleModuleComponent {
    void inject(MyFragment myFragment);
    void inject(MyService myService);
    void inject(MyController myController);
    void inject(MyActivity myActivity);
}
```

The two references modules GeneralPurposeModule and SpecificModule can then be implemented as follows:

GeneralPurposeModule.java

```

@Module
public class GeneralPurposeModule {
    @Provides
    @Singleton
    public Retrofit getRetrofit(PropertiesReader propertiesReader, RetrofitHeaderInterceptor
headerInterceptor){
        // 这里是逻辑...
        return retrofit;
    }

    @Provides
    @Singleton
    public PropertiesReader getPropertiesReader(){
        return new PropertiesReader();
    }

    @Provides
    @Singleton
    public RetrofitHeaderInterceptor getRetrofitHeaderInterceptor(){
        return new RetrofitHeaderInterceptor();
    }
}

```

SpecificModule.java

```

@Singleton
@Module
public class SpecificModule {
    @Provides @Singleton
    public RetrofitController getRetrofitController(Retrofit retrofit){
        RetrofitController retrofitController = new RetrofitController();
        retrofitController.setRetrofit(retrofit);
        return retrofitController;
    }

    @Provides @Singleton
    public MyService getMyService(RetrofitController retrofitController){
        MyService myService = new MyService();
        myService.setRetrofitController(retrofitController);
        return myService;
    }
}

```

在依赖注入阶段，组件将根据需要从两个模块中获取对象。

这种方法在模块化方面非常有用。在示例中，有一个通用模块用于实例化组件，如Retrofit对象（用于处理网络通信）和一个PropertiesReader（负责处理配置文件）。还有一个特定模块，负责实例化与该特定应用组件相关的特定控制器和服务类。

第112.5节：如何在build.gradle中添加Dagger 2

自Gradle 2.2发布以来，不再使用android-apt插件。应使用以下方法来设置Dagger 2。对于较旧版本的Gradle，请使用下面显示的旧方法。

对于Gradle >= 2.2

```

@Module
public class GeneralPurposeModule {
    @Provides
    @Singleton
    public Retrofit getRetrofit(PropertiesReader propertiesReader, RetrofitHeaderInterceptor
headerInterceptor){
        // Logic here...
        return retrofit;
    }

    @Provides
    @Singleton
    public PropertiesReader getPropertiesReader(){
        return new PropertiesReader();
    }

    @Provides
    @Singleton
    public RetrofitHeaderInterceptor getRetrofitHeaderInterceptor(){
        return new RetrofitHeaderInterceptor();
    }
}

```

SpecificModule.java

```

@Singleton
@Module
public class SpecificModule {
    @Provides @Singleton
    public RetrofitController getRetrofitController(Retrofit retrofit){
        RetrofitController retrofitController = new RetrofitController();
        retrofitController.setRetrofit(retrofit);
        return retrofitController;
    }

    @Provides @Singleton
    public MyService getMyService(RetrofitController retrofitController){
        MyService myService = new MyService();
        myService.setRetrofitController(retrofitController);
        return myService;
    }
}

```

During the dependency injection phase, the component will take objects from both modules according to the needs.

This approach is very useful in terms of *modularity*. In the example, there is a general purpose module used to instantiate components such as the Retrofit object (used to handle the network communication) and a PropertiesReader (in charge of handling configuration files). There is also a specific module that handles the instantiation of specific controllers and service classes in relation to that specific application component.

Section 112.5: How to add Dagger 2 in build.gradle

Since the release of Gradle 2.2, the use of the android-apt plugin is no longer used. The following method of setting up Dagger 2 should be used. For older version of Gradle, use the previous method shown below.

For Gradle >= 2.2

```

dependencies {
    // apt命令来自android-apt插件
    annotationProcessor 'com.google.dagger:dagger-compiler:2.8'
        compile 'com.google.dagger:dagger:2.8'
    provided 'javax.annotation:jsr250-api:1.0'
}

```

对于 Gradle 版本低于 2.2

要使用 Dagger 2，必须添加 android-apt 插件，在根目录的 build.gradle 中添加：

```

buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
            classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}

```

然后应用模块的 build.gradle 应包含：

```

apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'

android {
    ...
}

final DAGGER_VERSION = '2.0.2'
dependencies {
    ...

compile "com.google.dagger:dagger:${DAGGER_VERSION}"
    apt "com.google.dagger:dagger-compiler:${DAGGER_VERSION}"
}

```

参考资料：https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

第112.6节：构造函数注入

没有依赖项的类可以由 Dagger 轻松创建。

```

public class 引擎 {
    @Inject // <- 注释你的构造函数。
    public 引擎() {
    }
}

```

该类可以由任何组件提供。它本身没有依赖项且没有作用域。无需进一步代码。

依赖项作为构造函数的参数声明。只要这些依赖项可以被提供，Dagger 将调用构造函数并提供依赖项。

```

public class 汽车 {

```

```

dependencies {
    // apt command comes from the android-apt plugin
    annotationProcessor 'com.google.dagger:dagger-compiler:2.8'
        compile 'com.google.dagger:dagger:2.8'
    provided 'javax.annotation:jsr250-api:1.0'
}

```

For Gradle < 2.2

To use Dagger 2 it's necessary to add android-apt plugin, add this to the root build.gradle:

```

buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}

```

Then the application module's build.gradle should contain:

```

apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'

android {
    ...
}

final DAGGER_VERSION = '2.0.2'
dependencies {
    ...

compile "com.google.dagger:dagger:${DAGGER_VERSION}"
    apt "com.google.dagger:dagger-compiler:${DAGGER_VERSION}"
}

```

Reference: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Section 112.6: Constructor Injection

Classes without dependencies can easily be created by dagger.

```

public class Engine {
    @Inject // <- Annotate your constructor.
    public Engine() {
    }
}

```

This class can be provided by *any* component. It has *no dependencies itself* and is *not scoped*. There is no further code necessary.

Dependencies are declared as parameters in the constructor. Dagger will call the constructor and supply the dependencies, as long as those dependencies can be provided.

```

public class Car {

```

```
private 引擎 引擎;  
  
@Inject  
public Car(Engine engine) {  
    this.engine = engine;  
}  
}
```

只有当组件也能提供其所有依赖项——在本例中为 Engine——时，每个组件都可以提供此类。由于 Engine 也可以通过构造函数注入，任何组件都可以提供一个 Car。

当组件能够提供所有依赖项时，可以使用构造函数注入。一个组件可以提供一个依赖项，如果

- 它可以通过构造函数注入创建该依赖项
- 组件的一个模块可以提供该依赖项
- 它可以由父组件提供（如果它是一个 @Subcomponent）
- 它可以使用其依赖的组件暴露的对象（组件依赖）

```
private Engine engine;  
  
@Inject  
public Car(Engine engine) {  
    this.engine = engine;  
}  
}
```

This class can be provided by every component *iff* this component can also provide all of its dependencies—Engine in this case. Since Engine can also be constructor injected, *any* component can provide a Car.

You can use constructor injection whenever all of the dependencies can be provided by the component. A component can provide a dependency, if

- it can create it by using constructor injection
- a module of the component can provide it
- it can be provided by the parent component (if it is a @Subcomponent)
- it can use an object exposed by a component it depends on (component dependencies)

第113章：Realm

Realm 移动数据库是 SQLite 的替代方案。Realm 移动数据库比 ORM 快得多，通常也比原生 SQLite 快。

优点

离线功能、快速查询、安全线程、跨平台应用、加密、响应式架构。

第113.1节：排序查询

为了对查询结果进行排序，不应使用`findAll()`，而应使用`findAllSorted()`。

```
RealmResults<SomeObject> 结果 = realm.where(SomeObject.class)
    .findAllSorted("sortField", Sort.ASCENDING);
```

注意：

`sort()` 返回一个全新的已排序的 `RealmResults`，但对该 `RealmResults` 的更新会重置排序。如果你使用 `sort()`，应该始终在你的 `RealmChangeListener` 中重新排序，先从之前的 `RealmResults` 中移除 `RealmChangeListener`，然后添加到返回的新 `RealmResults` 上。在尚未加载的异步查询返回的 `RealmResults` 上使用 `sort()` 会失败。

`findAllSorted()` 会始终返回按字段排序的结果，即使结果被更新。推荐使用 `findAllSorted()`。

第113.2节：在 RxJava 中使用 Realm

对于查询，Realm 提供了 `realmResults.asObservable()` 方法。只有在循环线程（通常是 UI 线程）上才能观察结果。

为使其工作，你的配置必须包含以下内容

```
realmConfiguration = new RealmConfiguration.Builder(context) // ...
    .rxFactory(new RealmObservableFactory()) //
    ...
.build();
```

之后，您可以将结果用作可观察对象。

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

对于异步查询，您应通过`isLoaded()`进行过滤，这样只有在查询执行后才会收到事件。对于同步查询不需要此`filter()`，因为同步查询中`isLoaded()`总是返回`true`。

```
Subscription subscription = RxTextView.textChanges(editText).switchMap(charSequence ->
    realm.where(SomeObject.class)
        .contains("searchField", charSequence.toString(), Case.INSENSITIVE)
        .findAllAsync()
        .asObservable()
        .filter(RealmResults::isLoaded) //
        .subscribe(objects -> adapter.updateData(objects));
```

Chapter 113: Realm

Realm Mobile Database is an alternative to SQLite. Realm Mobile Database is much faster than an ORM, and often faster than raw SQLite.

Benefits

Offline functionality, Fast queries, Safe threading, Cross-platform apps, Encryption, Reactive architecture.

Section 113.1: Sorted queries

In order to sort a query, instead of using `findAll()`, you should use `findAllSorted()`.

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
    .findAllSorted("sortField", Sort.ASCENDING);
```

Note:

`sort()` returns a completely new `RealmResults` that is sorted, but an update to this `RealmResults` will reset it. If you use `sort()`, you should always re-sort it in your `RealmChangeListener`, remove the `RealmChangeListener` from the previous `RealmResults` and add it to the returned new `RealmResults`. Using `sort()` on a `RealmResults` returned by an async query that is not yet loaded will fail.

`findAllSorted()` will always return the results sorted by the field, even if it gets updated. It is recommended to use `findAllSorted()`.

Section 113.2: Using Realm with RxJava

For queries, Realm provides the `realmResults.asObservable()` method. Observing results is only possible on looper threads (typically the UI thread).

For this to work, your configuration must contain the following

```
realmConfiguration = new RealmConfiguration.Builder(context) // ...
    .rxFactory(new RealmObservableFactory()) //
    ...
.build();
```

Afterwards, you can use your results as an observable.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

For asynchronous queries, you should filter the results by `isLoaded()`, so that you receive an event only when the query has been executed. This `filter()` is not needed for synchronous queries (`isLoaded()` always returns `true` on sync queries).

```
Subscription subscription = RxTextView.textChanges(editText).switchMap(charSequence ->
    realm.where(SomeObject.class)
        .contains("searchField", charSequence.toString(), Case.INSENSITIVE)
        .findAllAsync()
        .asObservable()
        .filter(RealmResults::isLoaded) //
        .subscribe(objects -> adapter.updateData(objects));
```

对于写操作，您应使用executeTransactionAsync()方法，或者在后台线程打开Realm实例，同步执行事务，然后关闭Realm实例。

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm -> realm.insertOrUpdate(response.objects));
            }
        });
}
```

第113.3节：基本用法

设置实例

要使用Realm，首先需要获取其实例。每个Realm实例对应磁盘上的一个文件。获取实例的最基本方法如下：

```
// 创建配置
RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(context).build();

// 获取realm实例
Realm realm = Realm.getInstance(realmConfiguration);
// 或者
Realm.setDefaultConfiguration(realmConfiguration);
Realm realm = Realm.getDefaultInstance();
```

方法Realm.getInstance()会在数据库文件尚未创建时创建该文件，否则打开该文件。RealmConfiguration对象控制Realm创建的所有方面——无论是inMemory()数据库，Realm文件的名称，是否在需要迁移时清除Realm，初始数据等。

请注意，对Realm.getInstance()的调用是引用计数的（每次调用都会增加计数器），当调用realm.close()时计数器会减少。

关闭实例

在后台线程上，一旦Realm实例不再使用（例如，事务完成且线程执行结束），关闭Realm实例非常重要。未能关闭后台线程上的所有Realm实例会导致版本固定，并可能引起文件大小大幅增长。

```
Runnable runnable = new Runnable() {
    Realm realm = null;
    try {
        realm = Realm.getDefaultInstance();
        // ...
    } 最后 {
        if(realm != null) {
            realm.close();
        }
    }
};

new Thread(runnable).start(); // 后台线程，类似于AsyncTask的`doInBackground()`
```

值得注意的是，在API等级19以上，你可以用以下代码替代上述代码：

For writes, you should either use the executeTransactionAsync() method, or open a Realm instance on the background thread, execute the transaction synchronously, then close the Realm instance.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm -> realm.insertOrUpdate(response.objects));
            }
        });
}
```

Section 113.3: Basic Usage

Setting up an instance

To use Realm you first need to obtain an instance of it. Each Realm instance maps to a file on disk. The most basic way to get an instance is as follows:

```
// Create configuration
RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(context).build();

// Obtain realm instance
Realm realm = Realm.getInstance(realmConfiguration);
// or
Realm.setDefaultConfiguration(realmConfiguration);
Realm realm = Realm.getDefaultInstance();
```

The method Realm.getInstance() creates the database file if it has not been created, otherwise opens the file. The RealmConfiguration object controls all aspects of how a Realm is created - whether it's an inMemory() database, name of the Realm file, if the Realm should be cleared if a migration is needed, initial data, etc.

Please note that calls to Realm.getInstance() are reference counted (each call increments a counter), and the counter is decremented when realm.close() is called.

Closing an instance

On background threads, it's very important to **close** the Realm instance(s) once it's no longer used (for example, transaction is complete and the thread execution ends). Failure to close all Realm instances on background thread results in version pinning, and can cause a large growth in file size.

```
Runnable runnable = new Runnable() {
    Realm realm = null;
    try {
        realm = Realm.getDefaultInstance();
        // ...
    } finally {
        if(realm != null) {
            realm.close();
        }
    }
};

new Thread(runnable).start(); // background thread, like `doInBackground()` of AsyncTask
```

It's worth noting that above API Level 19, you can replace this code with just this:

```
try(Realm realm = Realm.getDefaultInstance()) {  
    // ...  
}
```

模型

下一步是创建你的模型。这里可能会有人问，“什么是模型？”。模型是一个结构，定义了存储在数据库中的对象的属性。例如，下面我们对一本书进行建模。

```
public class Book extends RealmObject {  
  
    // 该实体的主键  
    @PrimaryKey  
    private long id;  
  
    private String title;  
  
    @Index // 更快的查询  
    private String author;  
  
    // 标准的 getter 和 setter  
    public long getId() {  
        return id;  
    }  
  
    public void setId(long id) {  
        this.id = id;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public void setTitle(String title) {  
        this.title = title;  
    }  
  
    public String getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(String author) {  
        this.author = author;  
    }  
}
```

请注意，您的模型应继承 `RealmObject` 类。主键也通过 `@PrimaryKey` 注解指定。主键可以为 `null`，但只有一个元素可以将 `null` 作为主键。您还可以对不应持久化到磁盘的字段使用 `@Ignore` 注解：

```
@Ignore  
private String isbn;
```

插入或更新数据

为了将书籍对象存储到您的 `Realm` 数据库实例中，您可以先创建模型的实例，然后通过 `copyToRealm` 方法将其存储到数据库。对于创建或更新，您可以使用 `copyToRealmOrUpdate`。（一个更快的替代方法是新添加的 `insertOrUpdate()`）。

```
try(Realm realm = Realm.getDefaultInstance()) {  
    // ...  
}
```

Models

Next step would be creating your models. Here a question might be asked, "what is a model?". A model is a structure which defines properties of an object being stored in the database. For example, in the following we model a book.

```
public class Book extends RealmObject {  
  
    // Primary key of this entity  
    @PrimaryKey  
    private long id;  
  
    private String title;  
  
    @Index // faster queries  
    private String author;  
  
    // Standard getters & setters  
    public long getId() {  
        return id;  
    }  
  
    public void setId(long id) {  
        this.id = id;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public void setTitle(String title) {  
        this.title = title;  
    }  
  
    public String getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(String author) {  
        this.author = author;  
    }  
}
```

Note that your models should extend `RealmObject` class. Primary key is also specified by `@PrimaryKey` annotation. Primary keys can be null, but only one element can have `null` as a primary key. Also you can use the `@Ignore` annotation for the fields that should not be persisted to the disk:

```
@Ignore  
private String isbn;
```

Inserting or updating data

In order to store a book object to your `Realm` database instance, you can first create an instance of your model and then store it to the database via `copyToRealm` method. For creating or updating you can use `copyToRealmOrUpdate`. (A faster alternative is the newly added `insertOrUpdate()`).

```
// 创建模型实例  
Book book = new Book();  
book.setId(1);  
book.setTitle("Walking on air");  
book.setAuthor("Taylor Swift")
```

```
// 存储到数据库  
realm.executeTransaction(new Realm.Transaction() {  
    @Override  
    public void execute(Realm realm) {  
        realm.insertOrUpdate(book);  
    }  
});
```

注意，所有对数据的更改必须在事务中进行。创建对象的另一种方式是使用以下模式：

```
Book book = realm.createObject(Book.class, primaryKey);  
...
```

查询数据库

- 所有书籍：

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- 所有ID大于10的书籍：

```
RealmResults<Book> results = realm.where(Book.class)  
    .greaterThan("id", 10)  
    .findAll();
```

- 作者为'Taylor Swift'或'%Peter%'的书籍：

```
RealmResults<Book> results = realm.where(Book.class)  
    .beginGroup()  
        .equalTo("author", "Taylor Swift")  
        .or()  
    .contains("author", "Peter")  
    .endGroup().findAll();
```

删除对象

例如，我们想删除所有Taylor Swift的书籍：

```
// 事务开始  
realm.executeTransaction(new Realm.Transaction() {  
    @Override  
    public void execute(Realm realm) {  
        // 第一步：查询所有泰勒·斯威夫特的书籍  
        RealmResults<Book> results = ...  
  
        // 第二步：删除Realm中的元素  
        results.deleteAllFromRealm();  
    }  
});
```

```
// Creating an instance of the model
```

```
Book book = new Book();  
book.setId(1);  
book.setTitle("Walking on air");  
book.setAuthor("Taylor Swift")  
  
// Store to the database  
realm.executeTransaction(new Realm.Transaction() {  
    @Override  
    public void execute(Realm realm) {  
        realm.insertOrUpdate(book);  
    }  
});
```

Note that all changes to data must happen in a transaction. Another way to create an object is using the following pattern:

```
Book book = realm.createObject(Book.class, primaryKey);  
...
```

Querying the database

- All books:

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- All books having id greater than 10:

```
RealmResults<Book> results = realm.where(Book.class)  
    .greaterThan("id", 10)  
    .findAll();
```

- Books by 'Taylor Swift' or '%Peter%':

```
RealmResults<Book> results = realm.where(Book.class)  
    .beginGroup()  
        .equalTo("author", "Taylor Swift")  
        .or()  
    .contains("author", "Peter")  
    .endGroup().findAll();
```

Deleting an object

For example, we want to delete all books by Taylor Swift:

```
// Start of transaction  
realm.executeTransaction(new Realm.Transaction() {  
    @Override  
    public void execute(Realm realm) {  
        // First Step: Query all Taylor Swift books  
        RealmResults<Book> results = ...  
  
        // Second Step: Delete elements in Realm  
        results.deleteAllFromRealm();  
    }  
});
```

第113.4节：原始类型列表（RealmList<Integer/String/...>）

Realm目前不支持存储原始类型列表。这在他们的待办事项中（GitHub issue #575），但在此期间，这里有一个解决方法。

为你的原始类型创建一个新类，这里使用的是Integer，但你可以根据需要存储的内容进行更改。

```
public class RealmInteger extends RealmObject {  
    private int val;  
  
    public RealmInteger() {}  
  
    public RealmInteger(int val) {  
        this.val = val;  
    }  
  
    // 获取器和设置器  
}
```

你现在可以在你的RealmObject中使用它。

```
public class MainObject extends RealmObject {  
  
    private String name;  
    private RealmList<RealmInteger> ints;  
  
    // 获取器和设置器  
}
```

如果你使用GSON来填充你的RealmObject，你需要添加一个自定义类型适配器。

```
Type token = new TypeToken<RealmList<RealmInteger>>(){}.getType();  
Gson gson = new GsonBuilder()  
.setExclusionStrategies(new ExclusionStrategy() {  
    @Override  
    public boolean shouldSkipField(FieldAttributes f) {  
        return f.getDeclaringClass().equals(RealmObject.class);  
    }  
  
    @Override  
    public boolean shouldSkipClass(Class<?> clazz) {  
        return false;  
    }  
})  
.registerTypeAdapter(token, new TypeAdapter<RealmList<RealmInteger>>() {  
  
    @Override  
    public void write(JsonWriter out, RealmList<RealmInteger> value) throws IOException {  
        // 空实现  
    }  
  
    @Override  
    public RealmList<RealmInteger> read(JsonReader in) throws IOException {  
        RealmList<RealmInteger> list = new RealmList<RealmInteger>();  
        in.beginArray();  
        while (in.hasNext()) {  
            list.add(new RealmInteger(in.nextInt()));  
        }  
        in.endArray();  
    }  
}).  
registerTypeAdapter(token, new TypeAdapter<RealmList<RealmInteger>>() {  
  
    @Override  
    public void write(JsonWriter out, RealmList<RealmInteger> value) throws IOException {  
        // 空实现  
    }  
  
    @Override  
    public RealmList<RealmInteger> read(JsonReader in) throws IOException {  
        RealmList<RealmInteger> list = new RealmList<RealmInteger>();  
        in.beginArray();  
        while (in.hasNext()) {  
            list.add(new RealmInteger(in.nextInt()));  
        }  
        in.endArray();  
    }  
}).
```

Section 113.4: List of primitives (RealmList<Integer/String/...>)

Realm currently does not support storing a list of primitives. It is on their todo list ([GitHub issue #575](#)), but for the meantime, here is a workaround.

Create a new class for your primitive type, this uses Integer, but change it for whatever you want to store.

```
public class RealmInteger extends RealmObject {  
    private int val;  
  
    public RealmInteger() {}  
  
    public RealmInteger(int val) {  
        this.val = val;  
    }  
  
    // Getters and setters  
}
```

You can now use this in your RealmObject.

```
public class MainObject extends RealmObject {  
  
    private String name;  
    private RealmList<RealmInteger> ints;  
  
    // Getters and setters  
}
```

If you are using GSON to populate your RealmObject, you will need to add a custom type adapter.

```
Type token = new TypeToken<RealmList<RealmInteger>>(){}.getType();  
Gson gson = new GsonBuilder()  
.setExclusionStrategies(new ExclusionStrategy() {  
    @Override  
    public boolean shouldSkipField(FieldAttributes f) {  
        return f.getDeclaringClass().equals(RealmObject.class);  
    }  
  
    @Override  
    public boolean shouldSkipClass(Class<?> clazz) {  
        return false;  
    }  
})  
.registerTypeAdapter(token, new TypeAdapter<RealmList<RealmInteger>>() {  
  
    @Override  
    public void write(JsonWriter out, RealmList<RealmInteger> value) throws IOException {  
        // Empty  
    }  
  
    @Override  
    public RealmList<RealmInteger> read(JsonReader in) throws IOException {  
        RealmList<RealmInteger> list = new RealmList<RealmInteger>();  
        in.beginArray();  
        while (in.hasNext()) {  
            list.add(new RealmInteger(in.nextInt()));  
        }  
        in.endArray();  
    }  
}).
```

```
    return list;
}
.create();
```

第113.5节：异步查询

每个同步查询方法（例如`findAll()`或`findAllSorted()`）都有一个异步对应方法（`findAllAsync()`/`findAllSortedAsync()`）。

异步查询将`RealmResults`的评估任务卸载到另一个线程。为了在当前线程接收这些结果，当前线程必须是一个`Looper`线程（即：异步查询通常只在UI线程上工作）。

```
RealmChangeListener<RealmResults<SomeObject>> realmChangeListener; // 字段变量

realmChangeListener = new RealmChangeListener<RealmResults<SomeObject>>() {
    @Override
    public void onChange(RealmResults<SomeObject> element) {
        // 异步结果现在已加载
        adapter.updateData(element);
    }
};

RealmResults<SomeObject> asyncResults = realm.where(SomeObject.class).findAllAsync();
asyncResults.addChangeListener(realmChangeListener);
```

第113.6节：将Realm添加到您的项目中

将以下依赖项添加到您的项目级别`build.gradle`文件中。

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

将以下内容添加到您的应用级别`build.gradle`文件的最顶部。

```
apply plugin: 'realm-android'
```

完成gradle同步后，Realm即作为依赖项添加到您的项目中！

Realm自2.0.0版本起使用前需要进行初始化调用。您可以在您的`Application`类中或第一个`Activity`的`onCreate`方法中完成此操作。

```
Realm.init(this); // 在 Realm 2.0.0 中新增
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

第113.7节：Realm模型

Realm模型必须继承`RealmObject`基类，它们定义了底层数据库的模式。

支持的字段类型有`boolean`、`byte`、`short`、`int`、`long`、`float`、`double`、`String`、`Date`、`byte[]`、指向其他`RealmObject`的链接，以及`RealmList<T extends RealmModel>`。

```
public class Person extends RealmObject {
```

```
    return list;
}
.create();
```

Section 113.5: Async queries

Every synchronous query method (such as `findAll()` or `findAllSorted()`) has an asynchronous counterpart (`findAllAsync()` / `findAllSortedAsync()`).

Asynchronous queries offload the evaluation of the `RealmResults` to another thread. In order to receive these results on the current thread, the current thread must be a looper thread (read: async queries typically only work on the UI thread).

```
RealmChangeListener<RealmResults<SomeObject>> realmChangeListener; // field variable

realmChangeListener = new RealmChangeListener<RealmResults<SomeObject>>() {
    @Override
    public void onChange(RealmResults<SomeObject> element) {
        // asyncResults are now loaded
        adapter.updateData(element);
    }
};

RealmResults<SomeObject> asyncResults = realm.where(SomeObject.class).findAllAsync();
asyncResults.addChangeListener(realmChangeListener);
```

Section 113.6: Adding Realm to your project

Add the following dependency to your **project** level `build.gradle` file.

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Add the following right at the top of your **app** level `build.gradle` file.

```
apply plugin: 'realm-android'
```

Complete a gradle sync and you now have Realm added as a dependency to your project!

Realm requires an initial call since 2.0.0 before using it. You can do this in your `Application` class or in your first `Activity`'s `onCreate` method.

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Section 113.7: Realm Models

[Realm models](#) must extend the `RealmObject` base class, they define the schema of the underlying database.

Supported field types are `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`, `String`, `Date`, `byte[]`, links to other `RealmObjects`, and `RealmList<T extends RealmModel>`.

```
public class Person extends RealmObject {
```

```

@PrimaryKey // 主键同时隐式为 @Index
    // 这是 `copyToRealmOrUpdate()` 更新对象所必需的。
private long id;

@Index // 索引使该字段的查询更快
@Required // 防止插入 `null` 值
private String name;

private RealmList<Dog> dogs; //--> 与 Dog 的多对多关系

private Person spouse; //--> 与 Person 的一对一关系

@Ignore
private Calendar birthday; // 不支持日历，但可以忽略

// getter 和 setter 方法
}

```

如果你向 RealmObject 添加（或移除）一个新字段，或者添加一个新的 RealmObject 类或删除现有的类，则需要进行迁移。你可以在 RealmConfiguration.Builder 中设置 deleteIfMigrationNeeded()，或者定义必要的迁移。当添加（或移除） @Required、@Index 或 @PrimaryKey 注解时，也需要进行迁移。

关系必须手动设置，它们不会基于主键自动建立。

从 0.88.0 版本开始，也可以在 RealmObject 类中使用公共字段代替私有字段/ getter/ setter。

如果类也使用了

```

@RealmClass
public class Person implements RealmModel {
    // ...
}

```

在这种情况下，像 person.deleteFromRealm() 或 person.addChangeListener() 这样的方法被替换为 RealmObject.deleteFromRealm(person) 和 RealmObject.addChangeListener(person)。

限制是，RealmObject 只能继承 RealmObject，且不支持 final、volatile 和 transient 字段。

重要的是，受管理的 RealmObject 类只能在事务中修改。受管理的 RealmObject 不能在线程之间传递。

第113.8节：try-with-resources

```

try (Realm realm = Realm.getDefaultInstance()) {
    realm.executeTransaction(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            //需要执行的事务操作
        }
    });
    //在 try-with-resources 中无需关闭 realm
}

```

try-with-resources 只能从 KITKAT (minSDK 19) 开始使用

```

@PrimaryKey // primary key is also implicitly an @Index
    // it is required for `copyToRealmOrUpdate()` to update the object.
private long id;

@Index // index makes queries faster on this field
@Required // prevents `null` value from being inserted
private String name;

private RealmList<Dog> dogs; //--> many relationship to Dog

private Person spouse; //--> one relationship to Person

@Ignore
private Calendar birthday; // calendars are not supported but can be ignored

// getters, setters
}

```

If you add (or remove) a new field to your RealmObject (or you add a new RealmObject class or delete an existing one), a **migration** will be needed. You can either set `deleteIfMigrationNeeded()` in your `RealmConfiguration.Builder`, or define the necessary migration. Migration is also required when adding (or removing) `@Required`, or `@Index`, or `@PrimaryKey` annotation.

Relationships must be set manually, they are NOT automatic based on primary keys.

Since 0.88.0, it is also possible to use public fields instead of private fields/getters/setters in RealmObject classes.

It is also possible to implement [RealmModel](#) instead of extending `RealmObject`, if the class is also annotated with `@RealmClass`.

```

@RealmClass
public class Person implements RealmModel {
    // ...
}

```

In that case, methods like `person.deleteFromRealm()` or `person.addChangeListener()` are replaced with `RealmObject.deleteFromRealm(person)` and `RealmObject.addChangeListener(person)`.

[Limitations](#) are that by a `RealmObject`, only `RealmObject` can be extended, and there is no support for `final`, `volatile` and `transient` fields.

It is important that a *managed* RealmObject class can only be modified in a transaction. A *managed* RealmObject cannot be passed between threads.

Section 113.8: try-with-resources

```

try (Realm realm = Realm.getDefaultInstance()) {
    realm.executeTransaction(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            //whatever Transaction that has to be done
        }
    });
    //No need to close realm in try-with-resources
}

```

The Try with resources can be used only from KITKAT (minSDK 19)

第114章：Android 版本

第114.1节：运行时检查设备上的 Android 版本

`Build.VERSION_CODES` 是当前已知的 SDK 版本代码的枚举。

为了根据设备的安卓版本有条件地运行代码，使用`TargetApi`注解以避免 Lint 错误，并在运行特定于 API 级别的代码之前检查构建版本。

以下是如何在支持低于 API 23 版本的项目中使用 API-23 引入的类的示例：

```
@Override  
@TargetApi(23)  
public void onResume() {  
    super.onResume();  
    if (android.os.Build.VERSION.SDK_INT <= Build.VERSION_CODES.M) {  
        //运行Marshmallow代码  
        FingerprintManager fingerprintManager = this.getSystemService(FingerprintManager.class);  
        //.....  
    }  
}
```

Chapter 114: Android Versions

Section 114.1: Checking the Android Version on device at runtime

`Build.VERSION_CODES` is an enumeration of the currently known SDK version codes.

In order to conditionally run code based on the device's Android version, use the `TargetApi` annotation to avoid Lint errors, and check the build version before running the code specific to the API level.

Here is an example of how to use a class that was introduced in API-23, in a project that supports API levels lower than 23:

```
@Override  
@TargetApi(23)  
public void onResume() {  
    super.onResume();  
    if (android.os.Build.VERSION.SDK_INT <= Build.VERSION_CODES.M) {  
        //run Marshmallow code  
        FingerprintManager fingerprintManager = this.getSystemService(FingerprintManager.class);  
        //.....  
    }  
}
```

第115章：Wi-Fi连接

第115.1节：使用WEP加密连接

此示例演示如何在给定SSID和密码的情况下连接到使用WEP加密的Wi-Fi接入点。

```
public boolean ConnectToNetworkWEP(String networkSSID, String password)
{
    try {
        WifiConfiguration conf = new WifiConfiguration();           conf.SSID =
        "\"" + networkSSID + "\""; // 请注意引号。字符串应包含带引号的SSID
        conf.wepKeys[0] = "\"" + password + "\""; // 先尝试带引号

        conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.OPEN);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.SHARED);

        WifiManager wifiManager = (WifiManager)
        this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        int networkId = wifiManager.addNetwork(conf);

        if (networkId == -1){
            // 如果是十六进制密码，尝试不带引号再试一次
            conf.wepKeys[0] = password;
            networkId = wifiManager.addNetwork(conf);
        }

        List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
        for( WifiConfiguration i : list ) {
            if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
                wifiManager.disconnect();
                wifiManager.enableNetwork(i.networkId, true);
                wifiManager.reconnect();
                break;
            }
        }

        // WiFi连接成功，返回true
        return true;
    } catch (Exception ex) {
        System.out.println(Arrays.toString(ex.getStackTrace()));
        return false;
    }
}
```

第115.2节：连接WPA2加密

此示例连接到使用WPA2加密的Wi-Fi接入点。

```
public boolean ConnectToNetworkWPA(String networkSSID, String password) {
    try {
        WifiConfiguration conf = new WifiConfiguration();           conf.SSID =
        "\"" + networkSSID + "\""; // 请注意引号。字符串应包含带引号的SSID
        conf.preSharedKey = "\"" + password + "\"";
        conf.status = WifiConfiguration.Status.ENABLED;
```

Chapter 115: Wi-Fi Connections

Section 115.1: Connect with WEP encryption

This example connects to a Wi-Fi access point with WEP encryption, given an SSID and the password.

```
public boolean ConnectToNetworkWEP(String networkSSID, String password)
{
    try {
        WifiConfiguration conf = new WifiConfiguration();
        conf.SSID = "\"" + networkSSID + "\""; // Please note the quotes. String should contain
        SSID in quotes
        conf.wepKeys[0] = "\"" + password + "\""; // Try it with quotes first

        conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.OPEN);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.SHARED);

        WifiManager wifiManager = (WifiManager)
        this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        int networkId = wifiManager.addNetwork(conf);

        if (networkId == -1){
            // Try it again with no quotes in case of hex password
            conf.wepKeys[0] = password;
            networkId = wifiManager.addNetwork(conf);
        }

        List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
        for( WifiConfiguration i : list ) {
            if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
                wifiManager.disconnect();
                wifiManager.enableNetwork(i.networkId, true);
                wifiManager.reconnect();
                break;
            }
        }

        // WiFi Connection success, return true
        return true;
    } catch (Exception ex) {
        System.out.println(Arrays.toString(ex.getStackTrace()));
        return false;
    }
}
```

Section 115.2: Connect with WPA2 encryption

This example connects to a Wi-Fi access point with WPA2 encryption.

```
public boolean ConnectToNetworkWPA(String networkSSID, String password) {
    try {
        WifiConfiguration conf = new WifiConfiguration();
        conf.SSID = "\"" + networkSSID + "\""; // Please note the quotes. String should contain SSID
        in quotes
        conf.preSharedKey = "\"" + password + "\"";
        conf.status = WifiConfiguration.Status.ENABLED;
```

```

conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);

Log.d("connecting", conf.SSID + " " + conf.preSharedKey);

WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
wifiManager.addNetwork(conf);

Log.d("连接后", conf.SSID + " " + conf.preSharedKey);

List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
for( WifiConfiguration i : list ) {
    if(i.SSID != null && i.SSID.equals("'" + networkSSID + "'")) {
        wifiManager.disconnect();
    }
    wifiManager.enableNetwork(i.networkId, true);
    wifiManager.reconnect();
    Log.d("重新连接", i.SSID + " " + conf.preSharedKey);

    break;
}

//WiFi 连接成功, 返回 true
return true;
} catch (Exception ex) {
    System.out.println(Arrays.toString(ex.getStackTrace()));
    return false;
}
}

```

第115.3节：扫描接入点

此示例扫描可用的接入点和临时网络。btnScan 激活由 WifiManager.startScan() 方法发起的扫描。扫描完成后，WifiManager 调用 SCAN_RESULTS_AVAILABLE_ACTION 意图，并由 WifiScanReceiver 类处理扫描结果。结果显示在 TextView 中。

```

public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    TextView txtWifiInfo;
    WifiManager wifi;
    WifiScanReceiver wifiReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wifi=(WifiManager) getSystemService(Context.WIFI_SERVICE);
        wifiReceiver = new WifiScanReceiver();

        txtWifiInfo = (TextView)findViewById(R.id.txtWifiInfo);
        Button btnScan = (Button)findViewById(R.id.btnScan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override

```

```

conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);

Log.d("connecting", conf.SSID + " " + conf.preSharedKey);

WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
wifiManager.addNetwork(conf);

Log.d("after connecting", conf.SSID + " " + conf.preSharedKey);

List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
for( WifiConfiguration i : list ) {
    if(i.SSID != null && i.SSID.equals("'" + networkSSID + "'")) {
        wifiManager.disconnect();
        wifiManager.enableNetwork(i.networkId, true);
        wifiManager.reconnect();
        Log.d("re connecting", i.SSID + " " + conf.preSharedKey);

        break;
    }
}

//WiFi Connection success, return true
return true;
} catch (Exception ex) {
    System.out.println(Arrays.toString(ex.getStackTrace()));
    return false;
}
}

```

Section 115.3: Scan for access points

This example scans for available access points and ad hoc networks. btnScan activates a scan initiated by the WifiManager.startScan() method. After the scan, WifiManager calls the SCAN_RESULTS_AVAILABLE_ACTION intent and the WifiScanReceiver class processes the scan result. The results are displayed in a TextView.

```

public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    TextView txtWifiInfo;
    WifiManager wifi;
    WifiScanReceiver wifiReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wifi=(WifiManager) getSystemService(Context.WIFI_SERVICE);
        wifiReceiver = new WifiScanReceiver();

        txtWifiInfo = (TextView)findViewById(R.id.txtWifiInfo);
        Button btnScan = (Button)findViewById(R.id.btnScan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override

```

```

    public void onClick(View v) {
        Log.i(TAG, "Start scan...");
        wifi.startScan();
    });
}

protected void onPause() {
    unregisterReceiver(wifiReceiver);
    super.onPause();
}

protected void onResume() {
registerReceiver(
    wifiReceiver,
    new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
);
super.onResume();
}

private class WifiScanReceiver extends BroadcastReceiver {
    public void onReceive(Context c, Intent intent) {
List<ScanResult> wifiScanList = wifi.getScanResults();
        txtWifiInfo.setText("");
        for(int i = 0; i < wifiScanList.size(); i++){String info = ((wifiScanList.get(i)).toString());
            txtWifiInfo.append(info+"\n");
        }
    }
}

```

权限

以下权限需要在AndroidManifest.xml中定义：

```

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />

```

android.permission.ACCESS_WIFI_STATE 是调用 WifiManager.getScanResults() 所必需的。没有 android.permission.CHANGE_WIFI_STATE 就无法使用 WifiManager.startScan() 发起扫描。

当项目编译目标为 api 级别 23 或更高 (Android 6.0 及以上) 时，必须插入 android.permission.ACCESS_FINE_LOCATION 或 android.permission.ACCESS_COARSE_LOCATION 权限。此外，该权限需要被请求，例如在主活动的 onCreate 方法中：

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
        Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}

```

```

    public void onClick(View v) {
        Log.i(TAG, "Start scan...");
        wifi.startScan();
    });
}

protected void onPause() {
    unregisterReceiver(wifiReceiver);
    super.onPause();
}

protected void onResume() {
registerReceiver(
    wifiReceiver,
    new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
);
super.onResume();
}

private class WifiScanReceiver extends BroadcastReceiver {
    public void onReceive(Context c, Intent intent) {
List<ScanResult> wifiScanList = wifi.getScanResults();
        txtWifiInfo.setText("");
        for(int i = 0; i < wifiScanList.size(); i++){
            String info = ((wifiScanList.get(i)).toString());
            txtWifiInfo.append(info+"\n\n");
        }
    }
}

```

Permissions

The following permissions need to be defined in *AndroidManifest.xml*:

```

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />

```

android.permission.ACCESS_WIFI_STATE is necessary for calling `WifiManager.getScanResults()`. Without android.permission.CHANGE_WIFI_STATE you cannot initiate a scan with `WifiManager.startScan()`.

When compiling the project for api level 23 or greater (Android 6.0 and up), either android.permission.ACCESS_FINE_LOCATION or android.permission.ACCESS_COARSE_LOCATION must be inserted. Furthermore that permission needs to be requested, e.g. in the onCreate method of your main activity:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
        Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}

```

第116章：传感器管理器 (SensorManager)

第116.1节：使用加速度计判断设备是否静止

将以下代码添加到onCreate()/onResume()方法中：

```
SensorManager sensorManager;
Sensor mAccelerometer;
final float movementThreshold = 0.5f; // 你可能需要更改此值。
boolean isMoving = false;
float[] prevValues = {1.0f, 1.0f, 1.0f};
float[] currValues = new float[3];

sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

您可能需要通过反复试验调整movementThreshold的灵敏度。然后，重写onSensorChanged()方法，如下所示：

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor == mAccelerometer) {
        System.arraycopy(event.values, 0, currValues, 0, event.values.length);
        if ((Math.abs(currValues[0] - prevValues[0]) > movementThreshold) ||
            (Math.abs(currValues[1] - prevValues[1]) > movementThreshold) ||
            (Math.abs(currValues[2] - prevValues[2]) > movementThreshold)) {
            isMoving = true;
        } else {
            isMoving = false;
        }
        System.arraycopy(currValues, 0, prevValues, 0, currValues.length);
    }
}
```

如果您想防止应用安装在没有加速度计的设备上，必须在清单文件中添加以下行：

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

第116.2节：获取传感器事件

从机载传感器获取传感器信息：

```
public class MainActivity extends Activity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor accelerometer;
    private Sensor gyroscope;

    float[] accelerometerData = new float[3];
    float[] gyroscopeData = new float[3];

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Chapter 116: SensorManager

Section 116.1: Decide if your device is static or not, using the accelerometer

Add the following code to the onCreate()/onResume() method:

```
SensorManager sensorManager;
Sensor mAccelerometer;
final float movementThreshold = 0.5f; // You may have to change this value.
boolean isMoving = false;
float[] prevValues = {1.0f, 1.0f, 1.0f};
float[] currValues = new float[3];

sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

You may have to adjust the sensitivity by adapting the movementThreshold by trial and error. Then, override the onSensorChanged() method as follows:

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor == mAccelerometer) {
        System.arraycopy(event.values, 0, currValues, 0, event.values.length);
        if ((Math.abs(currValues[0] - prevValues[0]) > movementThreshold) ||
            (Math.abs(currValues[1] - prevValues[1]) > movementThreshold) ||
            (Math.abs(currValues[2] - prevValues[2]) > movementThreshold)) {
            isMoving = true;
        } else {
            isMoving = false;
        }
        System.arraycopy(currValues, 0, prevValues, 0, currValues.length);
    }
}
```

If you want to prevent your app from being installed on devices that do not have an accelerometer, you have to add the following line to your manifest:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Section 116.2: Retrieving sensor events

Retrieving sensor information from the onboard sensors:

```
public class MainActivity extends Activity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor accelerometer;
    private Sensor gyroscope;

    float[] accelerometerData = new float[3];
    float[] gyroscopeData = new float[3];

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```

setContentView(R.layout.activity_main);

mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
gyroscope = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);

}

@Override
public void onResume() {
    //注册感兴趣传感器的监听器
    mSensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_FASTEST);
    mSensorManager.registerListener(this, gyroscope, SensorManager.SENSOR_DELAY_FASTEST);
    super.onResume();
}

@Override
protected void onPause() {
    //注销之前注册的所有监听器
    mSensorManager.unregisterListener(this);
    super.onPause();
}

@Override
public void onSensorChanged(SensorEvent event) {
    //检查被轮询的传感器数据类型并存储到对应的浮点数组中
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
accelerometerData = event.values;
    } else if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
        gyroscopeData = event.values;
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // TODO 自动生成的方法存根
}
}

```

第116.3节：传感器转换到世界坐标系

Android返回的传感器值是相对于手机坐标系的（例如，+Y指向手机顶部）。我们可以使用传感器管理器的旋转矩阵将这些传感器值转换到世界坐标系（例如，+Y指向磁北，切向地面）。

首先，您需要声明并初始化用于存储数据的矩阵/数组（例如，可以在onCreate方法中进行）：

```

float[] accelerometerData = new float[3];
float[] accelerometerWorldData = new float[3];
float[] gravityData = new float[3];
float[] magneticData = new float[3];
float[] rotationMatrix = new float[9];

```

接下来，我们需要检测传感器值的变化，将它们存储到相应的数组中（如果我们想稍后或在其他地方使用），然后计算旋转矩阵及其转换到世界坐标系的结果：

```

setContentView(R.layout.activity_main);

mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
gyroscope = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);

}

@Override
public void onResume() {
    //Register listeners for your sensors of interest
    mSensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_FASTEST);
    mSensorManager.registerListener(this, gyroscope, SensorManager.SENSOR_DELAY_FASTEST);
    super.onResume();
}

@Override
protected void onPause() {
    //Unregister any previously registered listeners
    mSensorManager.unregisterListener(this);
    super.onPause();
}

@Override
public void onSensorChanged(SensorEvent event) {
    //Check the type of sensor data being polled and store into corresponding float array
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        accelerometerData = event.values;
    } else if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
        gyroscopeData = event.values;
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // TODO Auto-generated method stub
}
}

```

Section 116.3: Sensor transformation to world coordinate system

The sensor values returned by Android are with respect to the phone's coordinate system (e.g. +Y points towards the top of the phone). We can transform these sensor values into a world coordinate system (e.g. +Y points towards magnetic North, tangential to the ground) using the sensor manager's rotation matrix

First, you would need to declare and initialize the matrices/arrays where data will be stored (you can do this in the onCreate method, for example):

```

float[] accelerometerData = new float[3];
float[] accelerometerWorldData = new float[3];
float[] gravityData = new float[3];
float[] magneticData = new float[3];
float[] rotationMatrix = new float[9];

```

Next, we need to detect changes in sensor values, store them into the corresponding arrays (if we want to use them later/elsewhere), then calculate the rotation matrix and resulting transformation into world coordinates:

```

public void onSensorChanged(SensorEvent event) {
    sensor = event.sensor;
    int i = sensor.getType();

    if (i == Sensor.TYPE_ACCELEROMETER) {
        accelerometerData = event.values;
    } else if (i == Sensor.TYPE_GRAVITY) {
        gravityData = event.values;
    } else if (i == Sensor.TYPE_MAGNETIC) {
        magneticData = event.values;
    }

    // 从重力和磁力传感器数据计算旋转矩阵
    SensorManager.getRotationMatrix(rotationMatrix, null, gravityData, magneticData);

    // 加速度的世界坐标系变换
    accelerometerWorldData[0] = rotationMatrix[0] * accelerometerData[0] + rotationMatrix[1] *
        accelerometerData[1] + rotationMatrix[2] * accelerometerData[2];
    accelerometerWorldData[1] = rotationMatrix[3] * accelerometerData[0] + rotationMatrix[4] *
        accelerometerData[1] + rotationMatrix[5] * accelerometerData[2];
    accelerometerWorldData[2] = rotationMatrix[6] * accelerometerData[0] + rotationMatrix[7] *
        accelerometerData[1] + rotationMatrix[8] * accelerometerData[2];
}

```

```

public void onSensorChanged(SensorEvent event) {
    sensor = event.sensor;
    int i = sensor.getType();

    if (i == Sensor.TYPE_ACCELEROMETER) {
        accelerometerData = event.values;
    } else if (i == Sensor.TYPE_GRAVITY) {
        gravityData = event.values;
    } else if (i == Sensor.TYPE_MAGNETIC) {
        magneticData = event.values;
    }

    //Calculate rotation matrix from gravity and magnetic sensor data
    SensorManager.getRotationMatrix(rotationMatrix, null, gravityData, magneticData);

    //World coordinate system transformation for acceleration
    accelerometerWorldData[0] = rotationMatrix[0] * accelerometerData[0] + rotationMatrix[1] *
        accelerometerData[1] + rotationMatrix[2] * accelerometerData[2];
        accelerometerWorldData[1] = rotationMatrix[3] * accelerometerData[0] + rotationMatrix[4] *
        accelerometerData[1] + rotationMatrix[5] * accelerometerData[2];
        accelerometerWorldData[2] = rotationMatrix[6] * accelerometerData[0] + rotationMatrix[7] *
        accelerometerData[1] + rotationMatrix[8] * accelerometerData[2];
}

```

第117章：进度条

第117.1节：Material线性进度条

根据Material文档：

线性进度指示器应始终从0%填充到100%，且数值绝不应减少。
它应以出现在标题或表单边缘的条形表示，并且会出现和消失。

要使用Material线性进度条，只需在你的xml中使用：

```
<ProgressBar  
    android:id="@+id/my_progressBar"  
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```



不确定进度

要创建不确定的进度条，请将`android:indeterminate`属性设置为`true`。

```
<ProgressBar
```

Chapter 117: ProgressBar

Section 117.1: Material Linear ProgressBar

According to [Material Documentation](#):

A linear progress indicator should always fill from 0% to 100% and never decrease in value.
It should be represented by bars on the edge of a header or sheet that appear and disappear.

To use a material Linear ProgressBar just use in your xml:

```
<ProgressBar  
    android:id="@+id/my_progressBar"  
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```



Indeterminate

To create **indeterminate** ProgressBar set the `android:indeterminate` attribute to `true`.

```
<ProgressBar
```

```
        android:id="@+id/my_progressBar"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminate="true"/>
```

确定进度

要创建确定的进度条，请将`android:indeterminate`属性设置为`false`，并使用`android:max`和`android:progress`属性：

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

只需使用此代码更新数值：

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

缓冲

要使用`ProgressBar`创建缓冲效果，请将`android:indeterminate`属性设置为`false`，并使用`android:max`、`android:progress`和`android:secondaryProgress`属性：

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"
    android:secondaryProgress="25"/>
```

缓冲值由`android:secondaryProgress`属性定义。

只需使用以下代码来更新数值：

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

不确定和确定

要获得这种类型的进度条，只需使用`android:indeterminate`

属性设置为`true`的不确定进度条。

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

然后当你需要从不确定进度切换到确定进度时，使用`setIndeterminate()`方法。

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

```
        android:id="@+id/my_progressBar"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminate="true"/>
```

Determinate

To create **determinate** `ProgressBar` set the `android:indeterminate` attribute to `false` and use the `android:max` and the `android:progress` attributes:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Just use this code to update the value:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Buffer

To create a **buffer** effect with the `ProgressBar` set the `android:indeterminate` attribute to `false` and use the `android:max`, the `android:progress` and the `android:secondaryProgress` attributes:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"
    android:secondaryProgress="25"/>
```

The buffer value is defined by `android:secondaryProgress` attribute.

Just use this code to update the values:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Indeterminate and Determinate

To obtain this kind of `ProgressBar` just use an indeterminate `ProgressBar` using the `android:indeterminate` attribute to `true`.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Then when you need to switch from indeterminate to determinate progress use `setIndeterminate()` method .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

第117.2节：为进度条着色

使用 AppCompat 主题时，ProgressBar 的颜色将是您定义的 colorAccent。

版本 ≥ 5.0

要在不更改强调色的情况下更改 ProgressBar 颜色，可以使用 android:theme 属性覆盖强调色：

```
<ProgressBar
    android:theme="@style/MyProgress"
    style="@style/Widget.AppCompat.ProgressBar" />

<!-- res/values/styles.xml -->
<style name="MyProgress" parent="Theme.AppCompat.Light">
    <item name="colorAccent">@color/myColor</item>
</style>
```

要为 ProgressBar 着色，可以在 xml 文件中使用属性 android:indeterminateTintMode 和 android:indeterminateTint

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

第117.3节：自定义进度条

CustomProgressBarActivity.java:

```
public class CustomProgressBarActivity extends AppCompatActivity {

    private TextView txtProgress;
    private ProgressBar progressBar;
    private int pStatus = 0;
    private Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_progressbar);

        txtProgress = (TextView) findViewById(R.id.txtProgress);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (pStatus <= 100) {
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            progressBar.setProgress(pStatus);
                            txtProgress.setText(pStatus + " %");
                        }
                    });
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {

```

Section 117.2: Tinting ProgressBar

Using an AppCompat theme, the ProgressBar's color will be the colorAccent you have defined.

Version ≥ 5.0

To change the ProgressBar color without changing the accent color you can use the android:theme attribute overriding the accent color:

```
<ProgressBar
    android:theme="@style/MyProgress"
    style="@style/Widget.AppCompat.ProgressBar" />

<!-- res/values/styles.xml -->
<style name="MyProgress" parent="Theme.AppCompat.Light">
    <item name="colorAccent">@color/myColor</item>
</style>
```

To tint the ProgressBar you can use in the xml file the attributes android:indeterminateTintMode and android:indeterminateTint

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Section 117.3: Customized progressbar

CustomProgressBarActivity.java:

```
public class CustomProgressBarActivity extends AppCompatActivity {

    private TextView txtProgress;
    private ProgressBar progressBar;
    private int pStatus = 0;
    private Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_progressbar);

        txtProgress = (TextView) findViewById(R.id.txtProgress);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (pStatus <= 100) {
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            progressBar.setProgress(pStatus);
                            txtProgress.setText(pStatus + " %");
                        }
                    });
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {

```

```

e.printStackTrace();
}
pStatus++;
}
}).start();
}

}

```

activity_custom_progressbar.xml:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.skholingua.android.custom_progressbar_circular.MainActivity" >

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_height="wrap_content">

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_centerInParent="true"
            android:indeterminate="false"
            android:max="100"
            android:progress="0"
            android:progressDrawable="@drawable/custom_progressbar_drawable"
            android:secondaryProgress="0" />

        <TextView
            android:id="@+id/txtProgress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/progressBar"
            android:layout_centerInParent="true"
            android:textAppearance="?android:attr/textAppearanceSmall" />
    </RelativeLayout>

</RelativeLayout>

```

custom_progressbar_drawable.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="-90"
    android:pivotX="50%"
    android:pivotY="50%">

```

```

e.printStackTrace();
}
pStatus++;
}
}).start();
}

}

```

activity_custom_progressbar.xml:

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.skholingua.android.custom_progressbar_circular.MainActivity" >

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_height="wrap_content">

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="250dp"
            android:layout_height="250dp"
            android:layout_centerInParent="true"
            android:indeterminate="false"
            android:max="100"
            android:progress="0"
            android:progressDrawable="@drawable/custom_progressbar_drawable"
            android:secondaryProgress="0" />

        <TextView
            android:id="@+id/txtProgress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/progressBar"
            android:layout_centerInParent="true"
            android:textAppearance="?android:attr/textAppearanceSmall" />
    </RelativeLayout>

</RelativeLayout>

```

custom_progressbar_drawable.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="-90"
    android:pivotX="50%"
    android:pivotY="50%">

```

```

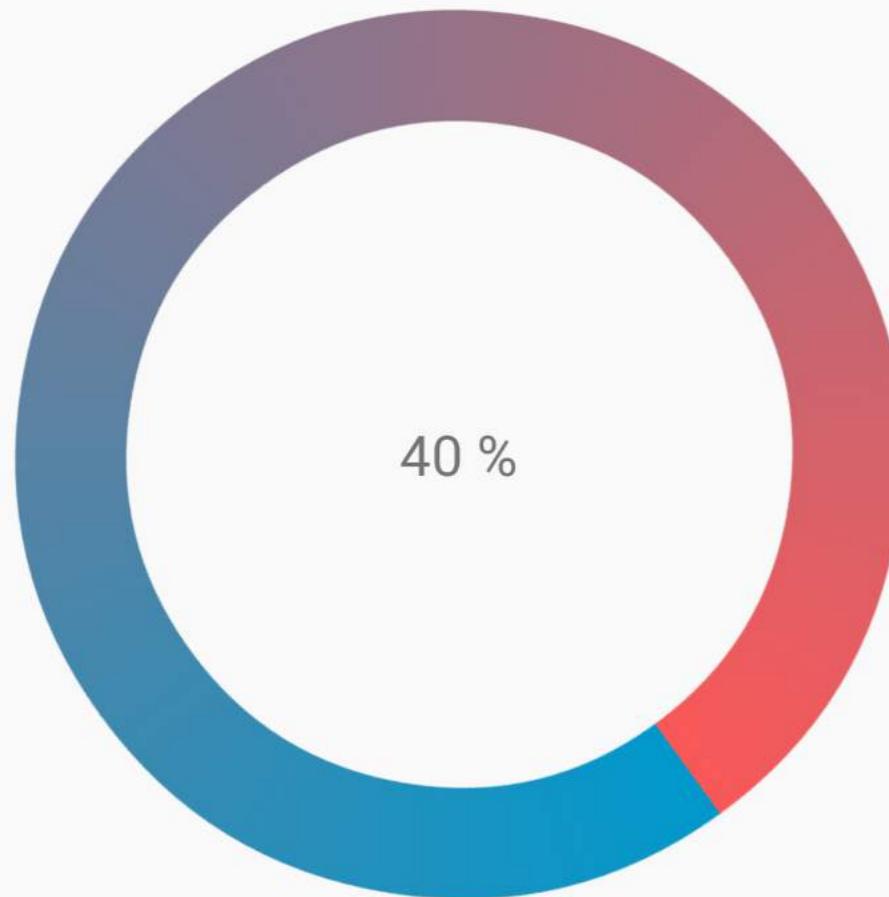
        android:toDegrees="270" >

<shape
    android:shape="ring"
    android:useLevel="false" >
<gradient
    android:centerY="0.5"
    android:endColor="#FA5858"
    android:startColor="#0099CC"
    android:type="sweep"
    android:useLevel="false" />

</rotate>

```

参考截图：



```

        android:toDegrees="270" >

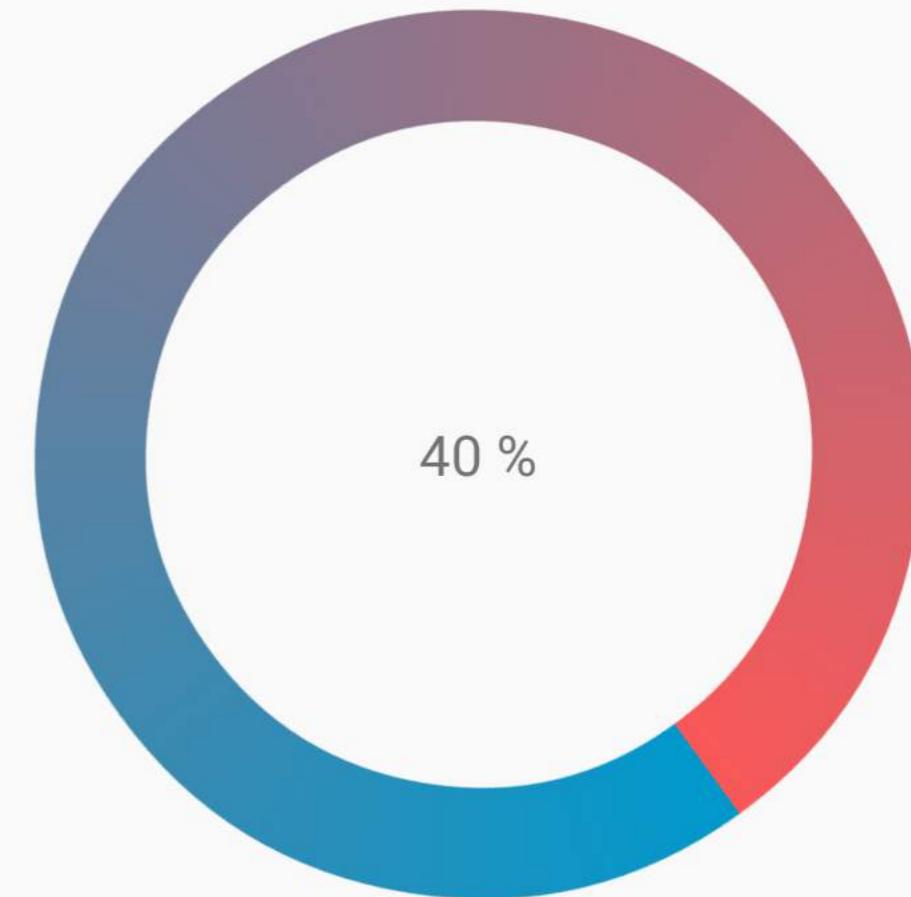
<shape
    android:shape="ring"
    android:useLevel="false" >
<gradient
    android:centerY="0.5"
    android:endColor="#FA5858"
    android:startColor="#0099CC"
    android:type="sweep"
    android:useLevel="false" />

</shape>

</rotate>

```

Reference screenshot:



第117.4节：创建自定义进度对话框

通过创建自定义进度对话框类，可以在UI实例中使用该对话框，而无需重新创建对话框。

首先创建一个自定义进度对话框类。

Section 117.4: Creating Custom Progress Dialog

By Creating Custom Progress Dialog class, the dialog can be used to show in UI instance, without recreating the dialog.

First Create a Custom Progress Dialog Class.

CustomProgress.java

```
public class CustomProgress {  
  
    public static CustomProgress customProgress = null;  
    private Dialog mDialog;  
  
    public static CustomProgress getInstance() {  
        if (customProgress == null) {  
            customProgress = new CustomProgress();  
        }  
        return customProgress;  
    }  
  
    public void showProgress(Context context, String message, boolean cancelable) {  
        mDialog = new Dialog(context);  
        // 对话框无标题  
        mDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);  
        mDialog.setContentView(R.layout.progress_bar_dialog);  
        mProgressBar = (ProgressBar) mDialog.findViewById(R.id.progress_bar);  
        // mProgressBar.getIndeterminateDrawable().setColorFilter(context.getResources()  
        // .getColor(R.color.material_blue_gray_500), PorterDuff.Mode.SRC_IN);  
        TextView progressText = (TextView) mDialog.findViewById(R.id.progress_text);  
        progressText.setText(" " + message);  
        progressText.setVisibility(View.VISIBLE);  
        mProgressBar.setVisibility(View.VISIBLE);  
        // 你可以根据需要更改或添加此行  
        mProgressBar.setIndeterminate(true);  
        mDialog.setCancelable(cancelable);  
        mDialog.setCanceledOnTouchOutside(cancelable);  
        mDialog.show();  
    }  
  
    public void hideProgress() {  
        if (mDialog != null) {  
            mDialog.dismiss();  
            mDialog = null;  
        }  
    }  
}
```

正在创建自定义进度布局

progress_bar_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="wrap_content"  
    android:layout_height="65dp"  
    android:background="@android:color/background_dark"  
    android:orientation="vertical">  
  
<TextView  
    android:id="@+id/progress_text"  
    android:layout_width="wrap_content"  
    android:layout_height="40dp"  
    android:layout_above="@+id/progress_bar"  
    android:layout_marginLeft="10dp"
```

CustomProgress.java

```
public class CustomProgress {  
  
    public static CustomProgress customProgress = null;  
    private Dialog mDialog;  
  
    public static CustomProgress getInstance() {  
        if (customProgress == null) {  
            customProgress = new CustomProgress();  
        }  
        return customProgress;  
    }  
  
    public void showProgress(Context context, String message, boolean cancelable) {  
        mDialog = new Dialog(context);  
        // no title for the dialog  
        mDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);  
        mDialog.setContentView(R.layout.progress_bar_dialog);  
        mProgressBar = (ProgressBar) mDialog.findViewById(R.id.progress_bar);  
        // mProgressBar.getIndeterminateDrawable().setColorFilter(context.getResources()  
        // .getColor(R.color.material_blue_gray_500), PorterDuff.Mode.SRC_IN);  
        TextView progressText = (TextView) mDialog.findViewById(R.id.progress_text);  
        progressText.setText(" " + message);  
        progressText.setVisibility(View.VISIBLE);  
        mProgressBar.setVisibility(View.VISIBLE);  
        // you can change or add this line according to your need  
        mProgressBar.setIndeterminate(true);  
        mDialog.setCancelable(cancelable);  
        mDialog.setCanceledOnTouchOutside(cancelable);  
        mDialog.show();  
    }  
  
    public void hideProgress() {  
        if (mDialog != null) {  
            mDialog.dismiss();  
            mDialog = null;  
        }  
    }  
}
```

Now creating the custom progress layout

progress_bar_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="wrap_content"  
    android:layout_height="65dp"  
    android:background="@android:color/background_dark"  
    android:orientation="vertical">  
  
<TextView  
    android:id="@+id/progress_text"  
    android:layout_width="wrap_content"  
    android:layout_height="40dp"  
    android:layout_above="@+id/progress_bar"  
    android:layout_marginLeft="10dp"
```

```

        android:layout_marginStart="10dp"
        android:background="@android:color/transparent"
        android:gravity="center_vertical"
        android:text=""
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:visibility="gone" />

<-- 样式可以更改为任何类型的进度条 -->

<ProgressBar
    android:id="@+id/progress_bar"
    style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_gravity="center"
    android:background="@color/cardview_dark_background"
    android:maxHeight="20dp"
    android:minHeight="20dp" />

</RelativeLayout>

```

就是这样。现在在代码中调用对话框

```

CustomProgress customProgress = CustomProgress.getInstance();

// 现在你已经有了 CustomProgress 的实例
// 用于显示进度条

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// 用于隐藏进度条

customProgress.hideProgress();

```

第117.5节：不确定进度条

不确定进度条显示一个循环动画，但不显示进度指示。

基本不确定进度条（旋转轮）

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

```

水平不确定进度条（扁平条）

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal" />

```

```

        android:layout_marginStart="10dp"
        android:background="@android:color/transparent"
        android:gravity="center_vertical"
        android:text=""
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:visibility="gone" />

<-- Where the style can be changed to any kind of ProgressBar -->

<ProgressBar
    android:id="@+id/progress_bar"
    style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_gravity="center"
    android:background="@color/cardview_dark_background"
    android:maxHeight="20dp"
    android:minHeight="20dp" />

</RelativeLayout>

```

This is it. Now for calling the Dialog in Code

```

CustomProgress customProgress = CustomProgress.getInstance();

// now you have the instance of CustomProgress
// for showing the ProgressBar

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// for hiding the ProgressBar

customProgress.hideProgress();

```

Section 117.5: Indeterminate ProgressBar

An indeterminate ProgressBar shows a cyclic animation without an indication of progress.

Basic indeterminate ProgressBar (spinning wheel)

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

```

Horizontal indeterminate ProgressBar (flat bar)

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal" />

```

其他内置进度条样式

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

在活动中使用不确定的进度条

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

第117.6节：确定进度条

确定进度条显示当前进度相对于特定最大值的进度。

水平确定进度条

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

垂直确定进度条

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res/drawable/progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@+id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@+id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
```

Other built-in ProgressBar styles

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

To use the indeterminate ProgressBar in an Activity

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Section 117.6: Determinate ProgressBar

A determinate ProgressBar shows the current progress towards a specific maximum value.

Horizontal determinate ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Vertical determinate ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res/drawable/progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@+id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@+id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
```

```

<corners android:radius="3dp"/>
<solid android:color="@android:color/holo_blue_dark"/>
</shape>
</clip>
</item>
</layer-list>

```

环形确定进度条

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>

```

res/drawable/progress_ring.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@+id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>

```

在活动中使用确定进度条 (determinate ProgressBar)。

```

ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);

```

```

<corners android:radius="3dp"/>
<solid android:color="@android:color/holo_blue_dark"/>
</shape>
</clip>
</item>
</layer-list>

```

Ring determinate ProgressBar

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>

```

res/drawable/progress_ring.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@+id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>

```

To use the determinate ProgressBar in an Activity.

```

ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);

```

第118章：自定义字体

第118.1节：画布文本中的自定义字体

使用assets中的字体在画布上绘制文本。

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

第118.2节：在Android O中使用字体

Android O改变了使用字体的方式。

Android O引入了一项新功能，称为XML中的字体，允许你将字体作为资源使用。这意味着，不再需要将字体打包为资产。字体现在被编译到R文件中，并自动作为系统资源可用。

为了添加新的字体，你需要执行以下操作：

- 创建一个新的资源目录：res/font。
- 将你的字体文件添加到该字体文件夹中。例如，添加myfont.ttf后，你将能够通过R.font.myfont使用该字体。

你还可以通过在res/font目录中添加以下XML文件来创建你自己的字体系列：

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font>
        android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/lobster_regular" />
    <font>
        android:fontStyle="italic"
        android:fontWeight="400"
        android:font="@font/lobster_italic" />
</font-family>
```

您可以以相同的方式使用font文件和font family文件：

- 在XML文件中，通过使用android:fontFamily属性，例如如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/myfont"/>
```

或者如下：

```
<style name="customfontstyle" parent="@android:style/TextAppearance.Small">
    <item name="android:fontFamily">@font/myfont</item>
</style>
```

Chapter 118: Custom Fonts

Section 118.1: Custom font in canvas text

Drawing text in canvas with your font from assets.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Section 118.2: Working with fonts in Android O

Android O changes the way to work with fonts.

Android O introduces a new feature, called *Fonts in XML*, which allows you to use fonts as resources. This means, that there is no need to bundle fonts as assets. Fonts are now compiled in an R file and are automatically available in the system as a resource.

In order to add a new **font**, you have to do the following:

- Create a new resource directory: res/font.
- Add your font files into this font folder. For example, by adding myfont.ttf, you will be able to use this font via R.font.myfont.

You can also create your own **font family** by adding the following XML file into the res/font directory:

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font>
        android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/lobster_regular" />
    <font>
        android:fontStyle="italic"
        android:fontWeight="400"
        android:font="@font/lobster_italic" />
</font-family>
```

You can use both the **font** file and the **font family** file in the same way:

- In an XML file, by using the android:fontFamily attribute, for example like this:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/myfont"/>
```

Or like this:

```
<style name="customfontstyle" parent="@android:style/TextAppearance.Small">
    <item name="android:fontFamily">@font/myfont</item>
</style>
```

- 在您的代码中，使用以下代码行：

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

第118.3节：自定义字体应用于整个活动

```
public class ReplaceFont {

    public static void changeDefaultFont(Context context, String oldFont, String assetsFont) {
        Typeface typeface = Typeface.createFromAsset(context.getAssets(), assetsFont);
        replaceFont(oldFont, typeface);
    }

    private static void replaceFont(String oldFont, Typeface typeface) {
        try {
            Field myField = Typeface.class.getDeclaredField(oldFont);
            myField.setAccessible(true);
            myField.set(null, typeface);
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}
```

然后在你的活动的 `onCreate()` 方法中：

```
// 将你的字体放入 assets 文件夹...
ReplaceFont.changeDefaultFont(getApplicationContext(), "DEFAULT", "LinLibertine.ttf");
```

第118.4节：在您的应用中使用自定义字体

1. 进入（项目文件夹）
2. 然后进入 `app -> src -> main`。
3. 在 `main` 文件夹中创建文件夹 '`assets -> fonts`'。
4. 将你的 '`fontfile.ttf`' 放入 `fonts` 文件夹中。

第118.5节：初始化字体

```
private Typeface myFont;

// 一个好的做法是在自定义的 Application 类的 onCreate() 中调用此方法// 并传入 'this' 作为 Context。只要应用存在，你的字体就可以使用

public void initFont(Context context) {
    myFont = Typeface.createFromAsset(context.getAssets(), "fonts/Roboto-Light.ttf");
}
```

第118.6节：在 TextView 中使用自定义字体

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

- In your code, by using the following lines of code:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Section 118.3: Custom font to whole activity

```
public class ReplaceFont {

    public static void changeDefaultFont(Context context, String oldFont, String assetsFont) {
        Typeface typeface = Typeface.createFromAsset(context.getAssets(), assetsFont);
        replaceFont(oldFont, typeface);
    }

    private static void replaceFont(String oldFont, Typeface typeface) {
        try {
            Field myField = Typeface.class.getDeclaredField(oldFont);
            myField.setAccessible(true);
            myField.set(null, typeface);
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}
```

Then in your activity, in `onCreate()` method:

```
// Put your font to assets folder...
ReplaceFont.changeDefaultFont(getApplicationContext(), "DEFAULT", "LinLibertine.ttf");
```

Section 118.4: Putting a custom font in your app

1. Go to the (project folder)
2. Then app -> src -> main.
3. Create folder 'assets -> fonts' into the main folder.
4. Put your 'fontfile.ttf' into the fonts folder.

Section 118.5: Initializing a font

```
private Typeface myFont;

// A good practice might be to call this in onCreate() of a custom
// Application class and pass 'this' as Context. Your font will be ready to use
// as long as your app lives
public void initFont(Context context) {
    myFont = Typeface.createFromAsset(context.getAssets(), "fonts/Roboto-Light.ttf");
}
```

Section 118.6: Using a custom font in a TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

第118.7节：通过xml为TextView应用字体（无需Java代码）

TextViewPlus.java :

```
public class TextViewPlus extends TextView {
    private static final String TAG = "TextView";

    public TextViewPlus(Context context) {
        super(context);
    }

    public TextViewPlus(Context context, AttributeSet attrs) {
        super(context, attrs);
        setCustomFont(context, attrs);
    }

    public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
        String customFont = a.getString(R.styleable.TextViewPlus_customFont);
        setCustomFont(ctx, customFont);
        a.recycle();
    }

    public boolean setCustomFont(Context ctx, String asset) {
        Typeface typeface = null;
        try {
            typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
        } catch (Exception e) {
            Log.e(TAG, "无法加载字体: "+e.getMessage());
            return false;
        }
        setTypeface(typeface);
        return true;
    }
}
```

attrs.xml: (放置在 res/values 目录下的位置)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

使用方法:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:foo="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

Section 118.7: Apply font on TextView by xml (Not required Java code)

TextViewPlus.java:

```
public class TextViewPlus extends TextView {
    private static final String TAG = "TextView";

    public TextViewPlus(Context context) {
        super(context);
    }

    public TextViewPlus(Context context, AttributeSet attrs) {
        super(context, attrs);
        setCustomFont(context, attrs);
    }

    public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
        String customFont = a.getString(R.styleable.TextViewPlus_customFont);
        setCustomFont(ctx, customFont);
        a.recycle();
    }

    public boolean setCustomFont(Context ctx, String asset) {
        Typeface typeface = null;
        try {
            typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
        } catch (Exception e) {
            Log.e(TAG, "Unable to load typeface: "+e.getMessage());
            return false;
        }
        setTypeface(typeface);
        return true;
    }
}
```

attrs.xml: (Where to place res/values)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

How to use:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:foo="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

```

<com.mypackage.TextViewPlus
    android:id="@+id/textViewPlus1"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:text="@string/showingOffTheNewTypeface"
    foo:customFont="my_font_name_regular.otf">
</com.mypackage.TextViewPlus>
</LinearLayout>

```

第118.8节：高效的字体加载

加载自定义字体可能导致性能下降。我强烈推荐使用这个小工具，它可以将你已经使用过的字体保存/加载到一个哈希表中。

```

public class TypefaceUtils {

private static final Hashtable<String, Typeface> sTypeFaces = new Hashtable<>();

/**
 * 从assets主目录通过文件名获取字体
 *
 * @param context
 * @param fileName 字体文件在assets主目录中的名称
 * @return
 */
public static Typeface getTypeFace(final Context context, final String fileName) {
    Typeface tempTypeface = sTypeFaces.get(fileName);

    if (tempTypeface == null) {
        tempTypeface = Typeface.createFromAsset(context.getAssets(), fileName);
        sTypeFaces.put(fileName, tempTypeface);
    }

    return tempTypeface;
}
}

```

用法：

```

Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);

```

```

<com.mypackage.TextViewPlus
    android:id="@+id/textViewPlus1"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:text="@string/showingOffTheNewTypeface"
    foo:customFont="my_font_name_regular.otf">
</com.mypackage.TextViewPlus>
</LinearLayout>

```

Section 118.8: Efficient Typeface loading

Loading custom fonts can be lead to a bad performance. I highly recommend to use this little helper which saves/loads your already used fonts into a Hashtable.

```

public class TypefaceUtils {

private static final Hashtable<String, Typeface> sTypeFaces = new Hashtable<>();

/**
 * Get typeface by filename from assets main directory
 *
 * @param context
 * @param fileName the name of the font file in the asset main directory
 * @return
 */
public static Typeface getTypeFace(final Context context, final String fileName) {
    Typeface tempTypeface = sTypeFaces.get(fileName);

    if (tempTypeface == null) {
        tempTypeface = Typeface.createFromAsset(context.getAssets(), fileName);
        sTypeFaces.put(fileName, tempTypeface);
    }

    return tempTypeface;
}
}

```

Usage:

```

Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);

```

第119章：获取系统字体名称及使用字体

以下示例展示了如何检索存储在 /system/fonts/ 目录中的系统字体默认名称，以及如何使用系统字体设置 TextView 元素的字体样式。

第119.1节：获取系统字体名称

```
ArrayList<String> fontNames = new ArrayList<String>();
File temp = new File("/system/fonts/");
String fontSuffix = ".ttf";

for(File font : temp.listFiles()) {
    String fontName = font.getName();
    if(fontName.endsWith(fontSuffix)) {
        fontNames.add(fontName.subSequence(0, fontName.lastIndexOf(fontSuffix)).toString());
    }
}
```

第119.2节：将系统字体应用于TextView

在以下代码中，您需要将fontsname替换为您想使用的字体名称：

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Chapter 119: Getting system font names and using the fonts

The following examples show how to retrieve the default names of the system fonts that are stored in the /system/fonts/ directory and how to use a system font to set the typeface of a TextView element.

Section 119.1: Getting system font names

```
ArrayList<String> fontNames = new ArrayList<String>();
File temp = new File("/system/fonts/");
String fontSuffix = ".ttf";

for(File font : temp.listFiles()) {
    String fontName = font.getName();
    if(fontName.endsWith(fontSuffix)) {
        fontNames.add(fontName.subSequence(0, fontName.lastIndexOf(fontSuffix)).toString());
    }
}
```

Section 119.2: Applying a system font to a TextView

In the following code you need to replace fontsname by the name of the font you would like to use:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

第120章：文本转语音 (TTS)

第120.1节：文本转语音基础

layout_text_to_speech.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="请输入文本！"
        android:id="@+id/textToSpeak"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textToSpeak"
        android:id="@+id/btnSpeak"/>

</RelativeLayout>
```

AndroidTextToSpeechActivity.java

```
public class AndroidTextToSpeechActivity extends Activity implements
    TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeak = findViewById(R.id.textToSpeak);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnSpeak.setEnabled(false);
        tts = new TextToSpeech(this, this);
        btnSpeak.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                speakOut();
            }
        });
    }

    @Override
    public void onDestroy() {
        // 别忘了关闭 tts !
        if (tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }
}
```

Chapter 120: Text to Speech(TTS)

Section 120.1: Text to Speech Base

layout_text_to_speech.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here!"
        android:id="@+id/textToSpeak"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textToSpeak"
        android:id="@+id/btnSpeak"/>

</RelativeLayout>
```

AndroidTextToSpeechActivity.java

```
public class AndroidTextToSpeechActivity extends Activity implements
    TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeak = findViewById(R.id.textToSpeak);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnSpeak.setEnabled(false);
        tts = new TextToSpeech(this, this);
        btnSpeak.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                speakOut();
            }
        });
    }

    @Override
    public void onDestroy() {
        // Don't forget to shutdown tts!
        if (tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }
}
```

```

@Override
public void onInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
        int result = tts.setLanguage(Locale.US);

        if (result == TextToSpeech.LANG_MISSING_DATA
            || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Log.e("TTS", "此语言不支持");
        } else {
            btnSpeak.setEnabled(true);
            speakOut();
        }
    } else {
        Log.e("TTS", "初始化失败 !");
    }
}

private void speakOut() {
    String text = textToSpeak.getText().toString();
    if(text == null || text.isEmpty())
        return;

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        String utteranceId=this.hashCode() + "";
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
    } else {
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
    }
}

```

可以通过向setLanguage()方法提供一个Locale来设置要使用的语言：

```
tts.setLanguage(Locale.CHINESE); // 中文语言
```

支持的语言数量因Android版本而异。可以使用isLanguageAvailable()方法来检查某种语言是否被支持：

```
tts.isLanguageAvailable(Locale.CHINESE);
```

可以使用setPitch()方法设置语音的音调水平。默认音调值为1.0。使用小于1.0的值降低音调，使用大于1.0的值提高音调：

```
tts.setPitch(0.6);
```

可以使用setSpeechRate()方法设置语速。默认语速为1.0。将其设置为2.0可以使语速加倍，设置为0.5可以使语速减半：

```
tts.setSpeechRate(2.0);
```

第120.2节：跨API的TextToSpeech实现

冷启动可观察实现，当TTS引擎完成朗读时发出true，订阅时开始朗读。

注意API级别21引入了不同的朗读方式：

```
public class RxTextToSpeech {
```

```

@Override
public void onInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
        int result = tts.setLanguage(Locale.US);

        if (result == TextToSpeech.LANG_MISSING_DATA
            || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Log.e("TTS", "This Language is not supported");
        } else {
            btnSpeak.setEnabled(true);
            speakOut();
        }
    } else {
        Log.e("TTS", "Initialization Failed!");
    }
}

private void speakOut() {
    String text = textToSpeak.getText().toString();
    if(text == null || text.isEmpty())
        return;

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        String utteranceId=this.hashCode() + "";
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
    } else {
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
    }
}

```

The language to be spoken can be set by providing a [Locale](#) to the [setLanguage\(\)](#) method:

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

The number of supported languages varies between Android levels. The method [isLanguageAvailable\(\)](#) can be used to check if a certain language is supported:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

The speech pitch level can be set by using the [setPitch\(\)](#) method. By default, the pitch value is 1.0. Use values less than 1.0 to decrease the pitch level or values greater than 1.0 to increase the pitch level:

```
tts.setPitch(0.6);
```

The speech rate can be set using [setSpeechRate\(\)](#). The default speech rate is 1.0. The speech rate can be doubled by setting it to 2.0 or made half by setting it to 0.5:

```
tts.setSpeechRate(2.0);
```

Section 120.2: TextToSpeech implementation across the APIs

Cold observable implementation, emits true when TTS engine finishes speaking, starts speaking when subscribed.

Notice that API level 21 introduces different way to perform speaking:

```
public class RxTextToSpeech {
```

```

    @Nullable RxTTSObservableOnSubscribe audio;

    WeakReference<Context> contextRef;

    public RxTextToSpeech(Context context) {
        this.contextRef = new WeakReference<>(context);
    }

    public void requestTTS(FragmentActivity activity, int requestCode) {
        Intent checkTTSIntent = new Intent();
        checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        activity.startActivityForResult(checkTTSIntent, requestCode);
    }

    public void cancelCurrent() {
        if (audio != null) {
            audio.dispose();
            audio = null;
        }
    }

    public Observable<Boolean> speak(String textToRead) {
        audio = new RxTTSObservableOnSubscribe(contextRef.get(), textToRead, Locale.GERMANY);
        return Observable.create(audio);
    }

    public static class RxTTSObservableOnSubscribe extends UtteranceProgressListener
        implements ObservableOnSubscribe<Boolean>,
        Disposable, Cancellable, TextToSpeech.OnInitListener {

        volatile boolean disposed;
        ObservableEmitter<Boolean> emitter;
        TextToSpeech textToSpeech;
        String text = "";
        Locale selectedLocale;
        上下文 context;

        public RxTTSObservableOnSubscribe(Context context, String text, Locale locale) {
            this.selectedLocale = locale;
            this.context = context;
            this.text = text;
        }

        @Override public void subscribe(ObservableEmitter<Boolean> e) throws Exception {
            this.emitter = e;
            if (context == null) {
                this.emitter.onError(new Throwable("nullable context, cannot execute " + text));
            } else {
                this.textToSpeech = new TextToSpeech(context, this);
            }
        }

        @Override @DebugLog public void dispose() {
            if (textToSpeech != null) {
                textToSpeech.setOnUtteranceProgressListener(null);
                textToSpeech.stop();
                textToSpeech.shutdown();
                textToSpeech = null;
            }
            disposed = true;
        }
    }
}

```

```

    @Nullable RxTTSObservableOnSubscribe audio;

    WeakReference<Context> contextRef;

    public RxTextToSpeech(Context context) {
        this.contextRef = new WeakReference<>(context);
    }

    public void requestTTS(FragmentActivity activity, int requestCode) {
        Intent checkTTSIntent = new Intent();
        checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        activity.startActivityForResult(checkTTSIntent, requestCode);
    }

    public void cancelCurrent() {
        if (audio != null) {
            audio.dispose();
            audio = null;
        }
    }

    public Observable<Boolean> speak(String textToRead) {
        audio = new RxTTSObservableOnSubscribe(contextRef.get(), textToRead, Locale.GERMANY);
        return Observable.create(audio);
    }

    public static class RxTTSObservableOnSubscribe extends UtteranceProgressListener
        implements ObservableOnSubscribe<Boolean>,
        Disposable, Cancellable, TextToSpeech.OnInitListener {

        volatile boolean disposed;
        ObservableEmitter<Boolean> emitter;
        TextToSpeech textToSpeech;
        String text = "";
        Locale selectedLocale;
        Context context;

        public RxTTSObservableOnSubscribe(Context context, String text, Locale locale) {
            this.selectedLocale = locale;
            this.context = context;
            this.text = text;
        }

        @Override public void subscribe(ObservableEmitter<Boolean> e) throws Exception {
            this.emitter = e;
            if (context == null) {
                this.emitter.onError(new Throwable("nullable context, cannot execute " + text));
            } else {
                this.textToSpeech = new TextToSpeech(context, this);
            }
        }

        @Override @DebugLog public void dispose() {
            if (textToSpeech != null) {
                textToSpeech.setOnUtteranceProgressListener(null);
                textToSpeech.stop();
                textToSpeech.shutdown();
                textToSpeech = null;
            }
            disposed = true;
        }
    }
}

```

```

@Override public boolean isDisposed() {
    return disposed;
}

@Override public void cancel() throws Exception {
    dispose();
}

@Override public void onInit(int status) {

    int languageCode = textToSpeech.setLanguage(selectedLocale);

    if (languageCode == android.speech.tts.TextToSpeech.LANG_COUNTRY_AVAILABLE) {
        textToSpeech.setPitch(1);
        textToSpeech.setSpeechRate(1.0f);
        textToSpeech.setOnUtteranceProgressListener(this);
        performSpeak();
    } else {
        emitter.onError(new Throwable("language " + selectedLocale.getCountry() + " is not
supported"));
    }
}

@Override public void onStart(String utteranceId) {
    //无操作
}

@Override public void onDone(String utteranceId) {
    this.emitter.onNext(true);
    this.emitter.onComplete();
}

@Override public void onError(String utteranceId) {
    this.emitter.onError(new Throwable("error TTS " + utteranceId));
}

void performSpeak() {

    if (isAtLeastApiLevel(21)) {
        speakWithNewApi();
    } else {
        speakWithOldApi();
    }
}

@RequiresApi(api = 21) void speakWithNewApi() {
    Bundle params = new Bundle();
    params.putString(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "");
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, params, uniqueId());
}

void speakWithOldApi() {
    HashMap<String, String> map = new HashMap<>();
    map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, uniqueId());
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, map);
}

private String uniqueId() {
    return UUID.randomUUID().toString();
}

```

```

@Override public boolean isDisposed() {
    return disposed;
}

@Override public void cancel() throws Exception {
    dispose();
}

@Override public void onInit(int status) {

    int languageCode = textToSpeech.setLanguage(selectedLocale);

    if (languageCode == android.speech.tts.TextToSpeech.LANG_COUNTRY_AVAILABLE) {
        textToSpeech.setPitch(1);
        textToSpeech.setSpeechRate(1.0f);
        textToSpeech.setOnUtteranceProgressListener(this);
        performSpeak();
    } else {
        emitter.onError(new Throwable("language " + selectedLocale.getCountry() + " is not
supported"));
    }
}

@Override public void onStart(String utteranceId) {
    //no-op
}

@Override public void onDone(String utteranceId) {
    this.emitter.onNext(true);
    this.emitter.onComplete();
}

@Override public void onError(String utteranceId) {
    this.emitter.onError(new Throwable("error TTS " + utteranceId));
}

void performSpeak() {

    if (isAtLeastApiLevel(21)) {
        speakWithNewApi();
    } else {
        speakWithOldApi();
    }
}

@RequiresApi(api = 21) void speakWithNewApi() {
    Bundle params = new Bundle();
    params.putString(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "");
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, params, uniqueId());
}

void speakWithOldApi() {
    HashMap<String, String> map = new HashMap<>();
    map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, uniqueId());
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, map);
}

private String uniqueId() {
    return UUID.randomUUID().toString();
}

```

```
public static boolean isAtLeastApiLevel(int apiLevel) {  
    return Build.VERSION.SDK_INT >= apiLevel;  
}
```

```
public static boolean isAtLeastApiLevel(int apiLevel) {  
    return Build.VERSION.SDK_INT >= apiLevel;  
}  
}
```

第121章：下拉选择框（Spinner）

第121.1节：基本的下拉选择框示例

下拉选择框（Spinner）是一种下拉输入类型。首先在布局中

```
<Spinner  
    android:id="@+id/spinner"      <!-- 用于在JAVA中引用此spinner的id--&gt;<br/>    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
</Spinner>
```

其次，在spinner中填充值，主要有两种方式来填充spinner的值。

1. 第一种是直接从XML创建，在res目录下的values文件夹中创建一个array.xml。创建此array

```
<string-array name="defaultValue">  
    <item>--选择城市区域--</item>  
    <item>--选择城市区域--</item>  
    <item>--选择城市区域--</item>  
</string-array>
```

现在在spinner的XML中添加这一行

```
        android:entries="@array/defaultValue"
```

2. 你也可以通过JAVA添加值

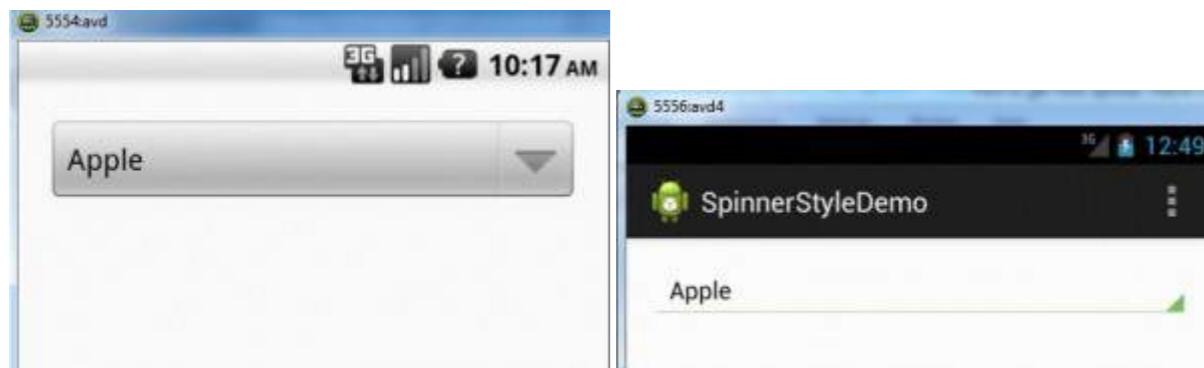
如果你在activity中使用，cityArea = (Spinner) findViewById(R.id.cityArea); 如果你在fragment中使用

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

现在创建一个字符串的arrayList

```
ArrayList<String> area = new ArrayList<>();  
//在area arrayList中添加值  
cityArea.setAdapter(new ArrayAdapter<String>(context  
        , android.R.layout.simple_list_item_1, area));
```

这将看起来像



根据设备的Android版本，它将呈现样式

Chapter 121: Spinner

Section 121.1: Basic Spinner Example

Spinner It is a type of dropdown input. Firstly in layout

```
<Spinner  
    android:id="@+id/spinner"      <!-- id to refer this spinner from JAVA--&gt;<br/>    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
</Spinner>
```

Now Secondly populate values in spinner There are mainly two ways to populate values in spinner.

1. From XML itself create a **array.xml** in **values** directory under **res**. Create this array

```
<string-array name="defaultValue">  
    <item>--Select City Area--</item>  
    <item>--Select City Area--</item>  
    <item>--Select City Area--</item>  
</string-array>
```

Now add this line in sppiner XML

```
        android:entries="@array/defaultValue"
```

2. You can also add values via JAVA

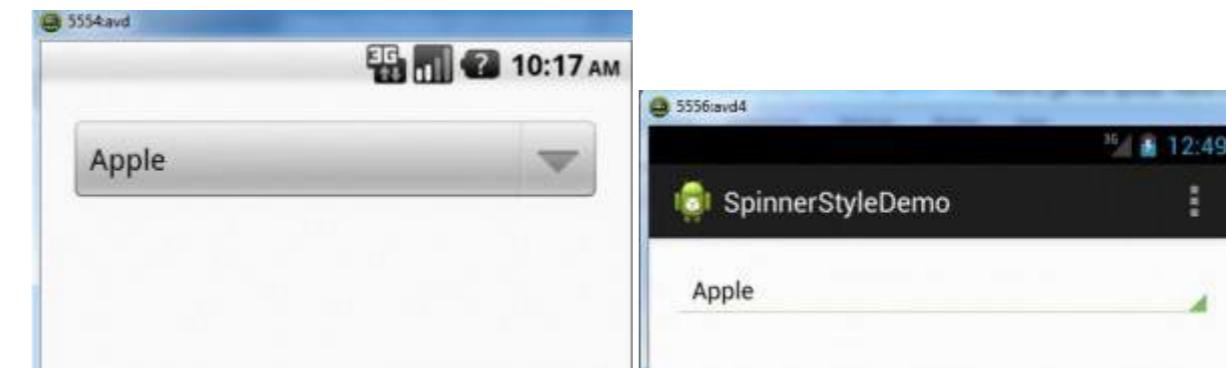
if you are using in activity cityArea = (Spinner) findViewById(R.id.cityArea); else if you are using in fragment

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Now create a arrayList of Strings

```
ArrayList<String> area = new ArrayList<>();  
//add values in area arrayList  
cityArea.setAdapter(new ArrayAdapter<String>(context  
        , android.R.layout.simple_list_item_1, area));
```

This will look like



According to the device Android version it will render style

以下是一些默认主题

如果应用程序未在其清单中明确请求主题，Android系统将根据应用程序的targetSdkVersion确定默认主题以保持应用程序的原始预期：

Android SDK 版本	默认主题
版本小于11	@android:style/Theme
版本介于11和13之间	@android:style/Theme.Holo
14及以上	@android:style/Theme.DeviceDefault

Spinner 可以通过xml轻松自定义，例如

```
android:background="@drawable/spinner_background"  
        android:layout_margin="16dp"  
        android:padding="16dp"
```

在XML中创建自定义背景并使用它。

轻松获取Spinner中选中项的位置及其他详情

```
cityArea.setOnItemSelectedListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        areaNo = position;  
    }  
  
    @Override  
    public void onNothingSelected(AdapterView<?> parent) {  
    }  
});
```

更改下拉列表中选中项的文本颜色

这可以通过两种方式在XML中完成

```
<item android:state_activated="true" android:color="@color/red"/>
```

这将更改弹出窗口中选中项的颜色。

并且在JAVA中这样做（在setOnItemSelectedListener(...)中）

```
@Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);  
        // 同样可以更改`背景颜色`等。  
    }
```

第121.2节：向您的活动添加下拉列表

在 /res/values/strings.xml 中：

```
<string-array name="spinner_options">  
    <item>选项 1</item>
```

Following are some of the default themes

If an app does not explicitly request a theme in its manifest, Android System will determine the default theme based on the app's targetSdkVersion to maintain the app's original expectations:

Android SDK Version	Default Theme
Version < 11	@android:style/Theme
Version between 11 and 13	@android:style/Theme.Holo
14 and higher	@android:style/Theme.DeviceDefault

Spinner 可以通过xml轻松自定义，例如

```
android:background="@drawable/spinner_background"  
        android:layout_margin="16dp"  
        android:padding="16dp"
```

Create a custom background in XML and use it.

easily get the position and other details of the selected item in spinner

```
cityArea.setOnItemSelectedListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        areaNo = position;  
    }  
  
    @Override  
    public void onNothingSelected(AdapterView<?> parent) {  
    }  
});
```

Change the text color of the selected item in spinner

This can be done in two ways in XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

This will change the selected item color in the popup.

and from JAVA do this (in the setOnItemSelectedListener(...))

```
@Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);  
        // similarly change `background color` etc.  
    }
```

Section 121.2: Adding a spinner to your activity

In /res/values/strings.xml:

```
<string-array name="spinner_options">  
    <item>Option 1</item>
```

```
<item>选项 2</item>
<item>选项 3</item>
</string-array>
```

在布局 XML 中：

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

在 Activity 中：

```
Spinner spinnerName = (Spinner) findViewById(R.id.spinnerName);
spinnerName.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String chosenOption = (String) parent.getItemAtPosition(position);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});
```

```
<item>Option 2</item>
<item>Option 3</item>
</string-array>
```

In layout XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

In Activity:

```
Spinner spinnerName = (Spinner) findViewById(R.id.spinnerName);
spinnerName.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String chosenOption = (String) parent.getItemAtPosition(position);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});
```

第122章：数据加密/解密

本主题讨论了Android中加密和解密的工作原理。

第122.1节：使用密码以安全方式进行数据的AES加密

下面的示例使用AES加密给定的数据块。加密密钥通过安全方式派生（随机盐，1000轮SHA-256）。加密使用CBC模式的AES，带有随机IV。

请注意，存储在类EncryptedData中的数据（salt、iv和encryptedData）可以连接成一个单一的字节数组。然后你可以保存数据或将其传输给接收方。

```
private static final int SALT_BYTES = 8;
private static final int PBK_ITERATIONS = 1000;
private static final String ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
private static final String PBE_ALGORITHM = "PBEwithSHA256and128BITAES-CBC-BC";

private EncryptedData encrypt(String password, byte[] data) throws NoSuchPaddingException,
NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException, InvalidAlgorithmParameterException {
    EncryptedData encData = new EncryptedData();
    SecureRandom rnd = new SecureRandom();
    encData.salt = new byte[SALT_BYTES];
    encData.iv = new byte[16]; // AES块大小
    rnd.nextBytes(encData.salt);
    rnd.nextBytes(encData.iv);

    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), encData.salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec ivSpec = new IvParameterSpec(encData.iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encData.encryptedData = cipher.doFinal(data);
    return encData;
}

private byte[] decrypt(String password, byte[] salt, byte[] iv, byte[] encryptedData) throws
NoSuchAlgorithmException, InvalidKeySpecException, NoSuchPaddingException, InvalidKeyException,
BadPaddingException, IllegalBlockSizeException, InvalidAlgorithmParameterException {
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
    return cipher.doFinal(encryptedData);
}

private static class EncryptedData {
    public byte[] salt;
    public byte[] iv;
    public byte[] encryptedData;
}
```

以下示例代码展示了如何测试加密和解密：

Chapter 122: Data Encryption/Decryption

This topic discusses how encryption and decryption works in Android.

Section 122.1: AES encryption of data using password in a secure way

The following example encrypts a given data block using [AES](#). The encryption key is derived in a secure way (random salt, 1000 rounds of SHA-256). The encryption uses AES in [CBC](#) mode with random [IV](#).

Note that the data stored in the class EncryptedData (salt, iv, and encryptedData) can be concatenated to a single byte array. You can then save the data or transmit it to the recipient.

```
private static final int SALT_BYTES = 8;
private static final int PBK_ITERATIONS = 1000;
private static final String ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
private static final String PBE_ALGORITHM = "PBEwithSHA256and128BITAES-CBC-BC";

private EncryptedData encrypt(String password, byte[] data) throws NoSuchPaddingException,
NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException, InvalidAlgorithmParameterException {
    EncryptedData encData = new EncryptedData();
    SecureRandom rnd = new SecureRandom();
    encData.salt = new byte[SALT_BYTES];
    encData.iv = new byte[16]; // AES block size
    rnd.nextBytes(encData.salt);
    rnd.nextBytes(encData.iv);

    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), encData.salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec ivSpec = new IvParameterSpec(encData.iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encData.encryptedData = cipher.doFinal(data);
    return encData;
}

private byte[] decrypt(String password, byte[] salt, byte[] iv, byte[] encryptedData) throws
NoSuchAlgorithmException, InvalidKeySpecException, NoSuchPaddingException, InvalidKeyException,
BadPaddingException, IllegalBlockSizeException, InvalidAlgorithmParameterException {
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
    return cipher.doFinal(encryptedData);
}

private static class EncryptedData {
    public byte[] salt;
    public byte[] iv;
    public byte[] encryptedData;
}
```

The following example code shows how to test encryption and decryption:

```
try {
    String password = "test12345";
    byte[] data = "plaintext11223344556677889900".getBytes("UTF-8");
    EncryptedData encData = encrypt(password, data);
    byte[] decryptedData = decrypt(password, encData.salt, encData.iv, encData.encryptedData);
    String decDataAsString = new String(decryptedData, "UTF-8");
    Toast.makeText(this, decDataAsString, Toast.LENGTH_LONG).show();
} catch (Exception e) {
    e.printStackTrace();
}
```

```
try {
    String password = "test12345";
    byte[] data = "plaintext11223344556677889900".getBytes("UTF-8");
    EncryptedData encData = encrypt(password, data);
    byte[] decryptedData = decrypt(password, encData.salt, encData.iv, encData.encryptedData);
    String decDataAsString = new String(decryptedData, "UTF-8");
    Toast.makeText(this, decDataAsString, Toast.LENGTH_LONG).show();
} catch (Exception e) {
    e.printStackTrace();
}
```

第123章：OkHttp

第123.1节：基本使用示例

我喜欢将我的OkHttp封装到一个名为HttpClient的类中，例如，在这个类里我为每个主要的HTTP动词提供方法，最常用的是post、get、put和delete。（我通常会包含一个接口，以便让它实现，从而能够在需要时轻松切换到不同的实现）：

```
public class HttpClient implements HttpClientInterface{  
  
    private static final String TAG = OkHttpClient.class.getSimpleName();  
    public static final MediaType JSON  
        = MediaType.parse("application/json; charset=utf-8");  
  
    OkHttpClient httpClient = new OkHttpClient();  
  
    @Override  
    public String post(String url, String json) throws IOException {  
        Log.i(TAG,  
"发送带有正文的post请求:" + json + " 到URL: " + url);  
        RequestBody body = RequestBody.create(JSON  
N, json);  
        Request request = new Request.Builder()  
            .url(url)  
.post(body)  
.build();  
        Response response = httpClient.newCall(request).execute();  
        return response.body().string();  
    }  
}
```

语法对于put、get和delete是相同的，除了一个词（.put(body)），所以发布那段代码可能会很烦人。用法非常简单，只需在某个url上调用相应的方法并传入一些json负载，方法将返回一个字符串结果，您可以稍后使用并解析。假设响应是一个json，我们可以很容易地从中创建一个JSONObject：

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);  
JSONObject json = new JSONObject(response);  
// 根据响应结构继续解析
```

第123.2节：设置OkHttp

通过 Maven 获取：

```
<dependency>  
    <groupId>com.squareup.okhttp3</groupId>  
    <artifactId>okhttp</artifactId>  
    <version>3.6.0</version>  
</dependency>
```

或使用 Gradle：

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

第 123.3 节：日志拦截器

拦截器用于拦截OkHttp调用。拦截的原因可能是为了监控、重写和重试

Chapter 123: OkHttp

Section 123.1: Basic usage example

I like to wrap my OkHttp into a class called HttpClient for example, and in this class I have methods for each of the major HTTP verbs, post, get, put and delete, most commonly. (I usually include an interface, in order to keep for it to implement, in order to be able to easily change to a different implementation, if need be):

```
public class HttpClient implements HttpClientInterface{  
  
    private static final String TAG = OkHttpClient.class.getSimpleName();  
    public static final MediaType JSON  
        = MediaType.parse("application/json; charset=utf-8");  
  
    OkHttpClient httpClient = new OkHttpClient();  
  
    @Override  
    public String post(String url, String json) throws IOException {  
        Log.i(TAG, "Sending a post request with body:\n" + json + "\n to URL: " + url);  
  
        RequestBody body = RequestBody.create(JSON, json);  
        Request request = new Request.Builder()  
            .url(url)  
            .post(body)  
            .build();  
        Response response = httpClient.newCall(request).execute();  
        return response.body().string();  
    }  
}
```

The syntax is the same for put, get and delete except for 1 word (.put(body)) so it might be obnoxious to post that code as well. Usage is pretty simple, just call the appropriate method on some url with some json payload and the method will return a string as a result that you can later use and parse. Let's assume that the response will be a json, we can create a JSONObject easily from it:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);  
JSONObject json = new JSONObject(response);  
// continue to parse the response according to its structure
```

Section 123.2: Setting up OkHttp

Grab via Maven:

```
<dependency>  
    <groupId>com.squareup.okhttp3</groupId>  
    <artifactId>okhttp</artifactId>  
    <version>3.6.0</version>  
</dependency>
```

or Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Section 123.3: Logging interceptor

Interceptors are used to intercept OkHttp calls. The reason to intercept could be to monitor, rewrite and retry

调用。它既可以用于发出的请求，也可以用于接收的响应。

```
class LoggingInterceptor implements Interceptor {
    @Override public Response intercept(Interceptor.Chain chain) throws IOException {
        Request request = chain.request();

        long t1 = System.nanoTime();
        logger.info(String.format("发送请求 %s 于 %s%n%s",
            request.url(), chain.connection(), request.headers()));

        Response response = chain.proceed(request);

        long t2 = System.nanoTime();
        logger.info(String.format("收到 %s 的响应，耗时 %.1f 毫秒%n%s",
            response.request().url(), (t2 - t1) / 1e6d, response.headers()));

        return response;
    }
}
```

第123.4节：同步Get调用

```
private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    Headers responseHeaders = response.headers();
    System.out.println(response.body().string());
}
```

第123.5节：异步Get调用

```
private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
        @Override public void onFailure(Call call, IOException e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

            Headers responseHeaders = response.headers();
            System.out.println(response.body().string());
        }
    });
}
```

calls. It can be used for outgoing request or incoming response both.

```
class LoggingInterceptor implements Interceptor {
    @Override public Response intercept(Interceptor.Chain chain) throws IOException {
        Request request = chain.request();

        long t1 = System.nanoTime();
        logger.info(String.format("Sending request %s on %s%n%s",
            request.url(), chain.connection(), request.headers()));

        Response response = chain.proceed(request);

        long t2 = System.nanoTime();
        logger.info(String.format("Received response for %s in %.1fms%n%s",
            response.request().url(), (t2 - t1) / 1e6d, response.headers()));

        return response;
    }
}
```

Section 123.4: Synchronous Get Call

```
private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    Headers responseHeaders = response.headers();
    System.out.println(response.body().string());
}
```

Section 123.5: Asynchronous Get Call

```
private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
        @Override public void onFailure(Call call, IOException e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

            Headers responseHeaders = response.headers();
            System.out.println(response.body().string());
        }
    });
}
```

```
});  
}
```

第123.6节：提交表单参数

```
private final OkHttpClient client = new OkHttpClient();  
  
public void run() throws Exception {  
    RequestBody formBody = new FormBody.Builder()  
        .add("search", "侏罗纪公园")  
.build();  
    Request request = new Request.Builder()  
        .url("https://en.wikipedia.org/w/index.php")  
        .post(formBody)  
.build();  
  
    Response response = client.newCall(request).execute();  
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);  
  
    System.out.println(response.body().string());  
}
```

```
});  
}
```

Section 123.6: Posting form parameters

```
private final OkHttpClient client = new OkHttpClient();  
  
public void run() throws Exception {  
    RequestBody formBody = new FormBody.Builder()  
        .add("search", "Jurassic Park")  
        .build();  
    Request request = new Request.Builder()  
        .url("https://en.wikipedia.org/w/index.php")  
        .post(formBody)  
        .build();  
  
    Response response = client.newCall(request).execute();  
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);  
  
    System.out.println(response.body().string());  
}
```

第123.7节：发送多部分请求

```
private static final String IMGUR_CLIENT_ID = "...";  
private static final MediaType MEDIA_TYPE_PNG = MediaType.parse("image/png");  
  
private final OkHttpClient client = new OkHttpClient();  
  
public void run() throws Exception {  
    // 使用 imgur 图片上传 API, 文档地址: https://api.imgur.com/endpoints/image  
    RequestBody requestBody = new MultipartBody.Builder()  
        .setType(MultipartBody.FORM)  
        .addFormDataPart("title", "Square Logo")  
        .addFormDataPart("image", "logo-square.png",  
            RequestBody.create(MEDIA_TYPE_PNG, new File("website/static/logo-square.png")))  
        .build();  
  
    Request request = new Request.Builder()  
        .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)  
        .url("https://api.imgur.com/3/image")  
.post(requestBody)  
        .build();  
  
    Response response = client.newCall(request).execute();  
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);  
  
    System.out.println(response.body().string());  
}
```

Section 123.7: Posting a multipart request

```
private static final String IMGUR_CLIENT_ID = "...";  
private static final MediaType MEDIA_TYPE_PNG = MediaType.parse("image/png");  
  
private final OkHttpClient client = new OkHttpClient();  
  
public void run() throws Exception {  
    // Use the imgur image upload API as documented at https://api.imgur.com/endpoints/image  
    RequestBody requestBody = new MultipartBody.Builder()  
        .setType(MultipartBody.FORM)  
        .addFormDataPart("title", "Square Logo")  
        .addFormDataPart("image", "logo-square.png",  
            RequestBody.create(MEDIA_TYPE_PNG, new File("website/static/logo-square.png")))  
        .build();  
  
    Request request = new Request.Builder()  
        .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)  
        .url("https://api.imgur.com/3/image")  
        .post(requestBody)  
        .build();  
  
    Response response = client.newCall(request).execute();  
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);  
  
    System.out.println(response.body().string());  
}
```

第123.8节：重写响应

```
private static final Interceptor REWRITE_CACHE_CONTROL_INTERCEPTOR = new Interceptor() {  
    @Override public Response intercept(Interceptor.Chain chain) throws IOException {  
        Response originalResponse = chain.proceed(chain.request());  
        return originalResponse.newBuilder()  
            .header("Cache-Control", "max-age=60")  
            .build();  
    }  
}
```

```
});  
}
```

Section 123.8: Rewriting Responses

```
private static final Interceptor REWRITE_CACHE_CONTROL_INTERCEPTOR = new Interceptor() {  
    @Override public Response intercept(Interceptor.Chain chain) throws IOException {  
        Response originalResponse = chain.proceed(chain.request());  
        return originalResponse.newBuilder()  
            .header("Cache-Control", "max-age=60")  
            .build();  
    }  
}
```

};

};

第124章：处理深度链接

<data> 属性	详细信息
方案	URI的scheme部分（区分大小写）。示例：http, https, ftp
主机	URI 的主机部分（区分大小写）。示例：google.com, example.org
端口	URI 的端口部分。示例：80, 443
路径	URI 的路径部分。必须以 / 开头。示例：/, /about
路径前缀	URI 路径部分的前缀。示例：/item, /article
路径模式	用于匹配 URI 路径部分的模式。示例：/item/*, /article/[0-9]*
Mime 类型	用于匹配的 Mime 类型。示例：image/jpeg, audio/*

深度链接是直接将用户带到您应用中特定内容的URL。您可以通过添加意图过滤器并从传入的意图中提取数据来设置深度链接，从而引导用户进入应用中的正确界面。

第124.1节：检索查询参数

```
public class MainActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        Intent intent = getIntent();  
        Uri data = intent.getData();  
  
        if (data != null) {  
            String param1 = data.getQueryParameter("param1");  
            String param2 = data.getQueryParameter("param2");  
        }  
    }  
}
```

如果用户点击链接 <http://www.example.com/map?param1=FOO¶m2=BAR>，那么这里的 param1 的值将是 "FOO"，param2 的值将是 "BAR"。

第124.2节：简单深度链接

AndroidManifest.xml：

```
<activity android:name="com.example.MainActivity" >  
  
    <intent-filter>  
        <action android:name="android.intent.action.VIEW" />  
        <category android:name="android.intent.category.DEFAULT" />  
        <category android:name="android.intent.category.BROWSABLE" />  
  
        <data android:scheme="http"  
              android:host="www.example.com" />  
    </intent-filter>  
  
</activity>
```

Chapter 124: Handling Deep Links

<data> Attribute	Details
scheme	The scheme part of a URI (case-sensitive). Examples: http, https, ftp
host	The host part of a URI (case-sensitive). Examples: google.com, example.org
port	The port part of a URI. Examples: 80, 443
path	The path part of a URI. Must begin with /. Examples: /, /about
pathPrefix	A prefix for the path part of a URI. Examples: /item, /article
pathPattern	A pattern to match for the path part of a URI. Examples: /item/*, /article/[0-9]*
mimeType	A mime type to match. Examples: image/jpeg, audio/*

Deep links are URLs that take users directly to specific content in your app. You can set up deep links by adding intent filters and extracting data from incoming intents to drive users to the right screen in your app.

Section 124.1: Retrieving query parameters

```
public class MainActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        Intent intent = getIntent();  
        Uri data = intent.getData();  
  
        if (data != null) {  
            String param1 = data.getQueryParameter("param1");  
            String param2 = data.getQueryParameter("param2");  
        }  
    }  
}
```

If the user clicks on a link to <http://www.example.com/map?param1=FOO¶m2=BAR>, then param1 here will have a value of "FOO" and param2 will have a value of "BAR".

Section 124.2: Simple deep link

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >  
  
    <intent-filter>  
        <action android:name="android.intent.action.VIEW" />  
        <category android:name="android.intent.category.DEFAULT" />  
        <category android:name="android.intent.category.BROWSABLE" />  
  
        <data android:scheme="http"  
              android:host="www.example.com" />  
    </intent-filter>  
  
</activity>
```

这将接受任何以http://www.example.com开头的链接，作为启动您的MainActivity的深度链接。

第124.3节：单一域名上的多个路径

AndroidManifest.xml：

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/about" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

当用户点击以下任意链接时，将启动您的MainActivity：

- http://www.example.com/
- http://www.example.com/about
- http://www.example.com/map

第124.4节：多个域名和多个路径

AndroidManifest.xml：

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:scheme="http"
              android:host="www.example2.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

当用户点击以下任意链接时，将启动您的 MainActivity：

- http://www.example.com/
- http://www.example2.com/

This will accept any link starting with http://www.example.com as a deep link to start your MainActivity.

Section 124.3: Multiple paths on a single domain

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/about" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any of these links:

- http://www.example.com/
- http://www.example.com/about
- http://www.example.com/map

Section 124.4: Multiple domains and multiple paths

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:scheme="http"
              android:host="www.example2.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any of these links:

- http://www.example.com/
- http://www.example2.com/

- <http://www.example.com/map>
- <http://www.example2.com/map>

第124.5节：同一域名的 http 和 https

AndroidManifest.xml :

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http" />
        <data android:scheme="https" />

        <data android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

当用户点击以下任意链接时，将启动您的 MainActivity：

- <http://www.example.com/>
- <https://www.example.com/>
- <http://www.example.com/map>
- <https://www.example.com/map>

第124.6节：使用 pathPrefix

AndroidManifest.xml :

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />

    </intent-filter>

</activity>
```

当用户点击任何以<http://www.example.com/item>开头的链接时，这将启动您的MainActivity，例如：

- <https://www.example.com/item>
- <http://www.example.com/item/1234>
- <https://www.example.com/item/xyz/details>

- <http://www.example.com/map>
- <http://www.example2.com/map>

Section 124.5: Both http and https for the same domain

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http" />
        <data android:scheme="https" />

        <data android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any of these links:

- <http://www.example.com/>
- <https://www.example.com/>
- <http://www.example.com/map>
- <https://www.example.com/map>

Section 124.6: Using pathPrefix

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any link starting with <http://www.example.com/item>, such as:

- <https://www.example.com/item>
- <http://www.example.com/item/1234>
- <https://www.example.com/item/xyz/details>

第125章：崩溃报告工具

第125.1节：Fabric - Crashlytics

Fabric是一个模块化的移动平台，提供了可组合使用的实用工具包来构建您的应用程序。Crashlytics是Fabric提供的一个崩溃和问题报告工具，允许您详细跟踪和监控您的应用程序。

如何配置Fabric-Crashlytics

步骤1：修改您的build.gradle文件：

添加插件仓库和gradle插件：

```
buildscript {  
    repositories {  
        maven { url 'https://maven.fabric.io/public' }  
    }  
  
    dependencies {  
        // Fabric Gradle 插件使用开放版本以快速响应 Android 工具更新  
        classpath 'io.fabric.tools:gradle:1.+'  
    }  
}
```

应用插件：

```
apply plugin: 'com.android.application'  
// 将 Fabric 插件放在 Android 插件之后  
apply plugin: 'io.fabric'
```

添加 Fabric 仓库：

```
repositories {  
    maven { url 'https://maven.fabric.io/public' }  
}
```

添加Crashlytics工具包：

```
dependencies {  
  
    compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {  
        transitive = true;  
    }  
}
```

步骤 2：在 AndroidManifest.xml 中添加您的 API Key 和 INTERNET 权限

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android">  
    <application  
        ...>  
  
        <meta-data  
            android:name="io.fabric.ApiKey"
```

Chapter 125: Crash Reporting Tools

Section 125.1: Fabric - Crashlytics

Fabric is a modular mobile platform that provides useful kits you can mix to build your application. **Crashlytics** is a crash and issue reporting tool provided by Fabric that allows you to track and monitor your applications in detail.

How to Configure Fabric-Crashlytics

Step 1: Change your build.gradle:

Add the plugin repo and the gradle plugin:

```
buildscript {  
    repositories {  
        maven { url 'https://maven.fabric.io/public' }  
    }  
  
    dependencies {  
        // The Fabric Gradle plugin uses an open ended version to react  
        // quickly to Android tooling updates  
        classpath 'io.fabric.tools:gradle:1.+'  
    }  
}
```

Apply the plugin:

```
apply plugin: 'com.android.application'  
// Put Fabric plugin after Android plugin  
apply plugin: 'io.fabric'
```

Add the Fabric repo:

```
repositories {  
    maven { url 'https://maven.fabric.io/public' }  
}
```

Add the Crashlytics Kit:

```
dependencies {  
  
    compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {  
        transitive = true;  
    }  
}
```

Step 2: Add Your **API Key** and the **INTERNET** permission in AndroidManifest.xml

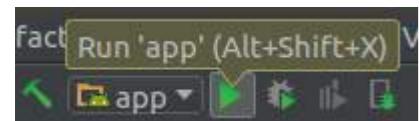
```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android">  
    <application  
        ...>  
  
        <meta-data  
            android:name="io.fabric.ApiKey"
```

```
    android:value="25eeaca3bb31cd41577e097cabd1ab9eee9da151d"  
/>  
  
<uses-permission android:name="android.permission.INTERNET" />  
</manifest>
```

步骤 3：在代码中运行时初始化该套件，例如：

```
public class MainActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        //初始化KIT  
Fabric.use(this, new Crashlytics());  
  
        setContentView(R.layout.activity_main);  
    }  
}
```

步骤4：构建项目。构建并运行：



使用Fabric IDE插件

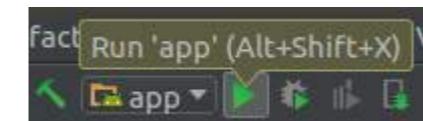
可以通过Fabric IDE插件为Android Studio或IntelliJ安装Kits，具体请参照此链接。[\[link\]](#)

```
    android:value="25eeaca3bb31cd41577e097cabd1ab9eee9da151d"  
/>  
  
</application>  
  
<uses-permission android:name="android.permission.INTERNET" />  
</manifest>
```

Step 3: Init the Kit at runtime in you code, for example:

```
public class MainActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        //Init the KIT  
Fabric.with(this, new Crashlytics());  
  
        setContentView(R.layout.activity_main);  
    }  
}
```

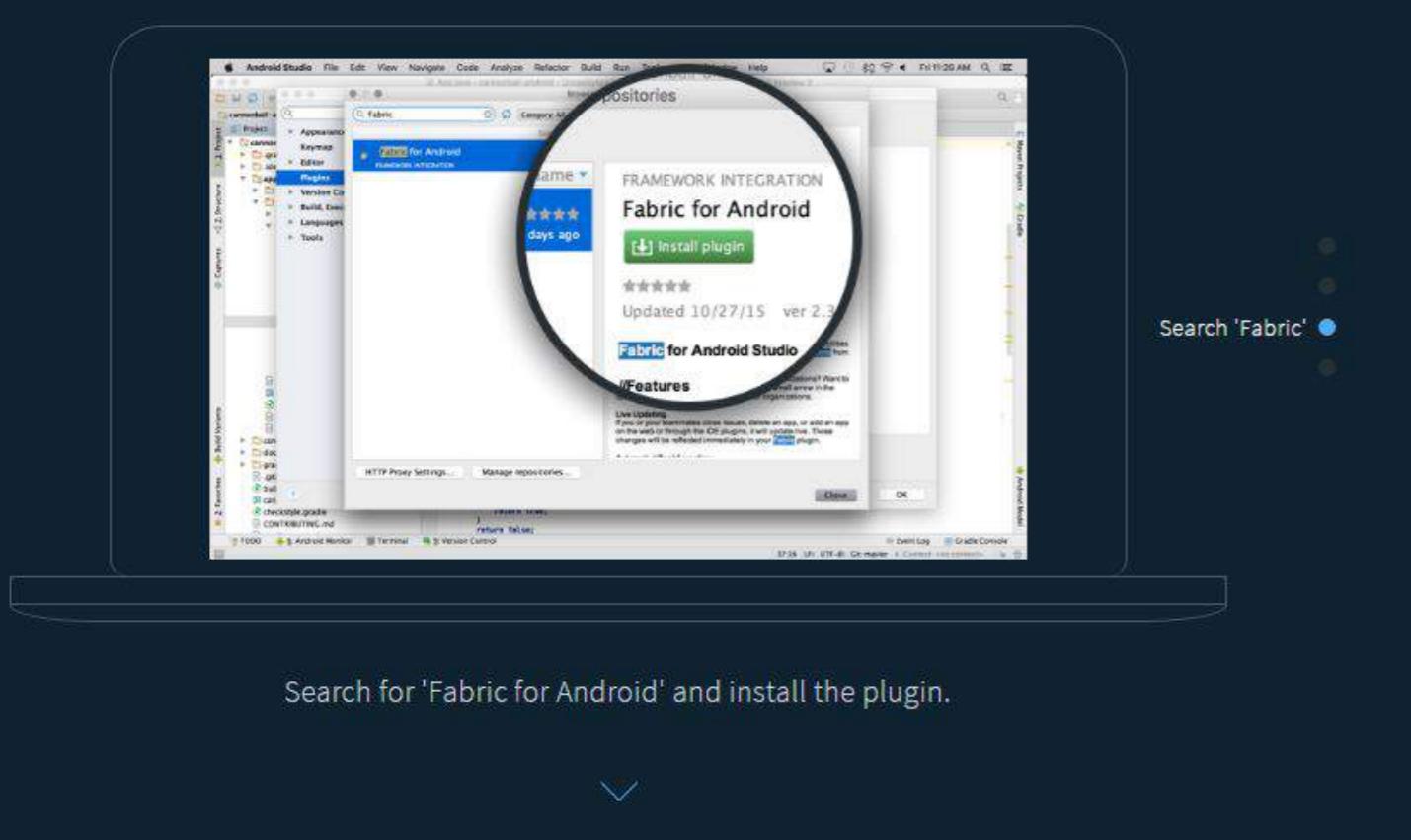
Step 4: Build project. To build and run:



Using the Fabric IDE plugin

Kits can be installed using the Fabric IDE plugin for Android Studio or IntelliJ following [this](#) link.

Android Studio / IntelliJ



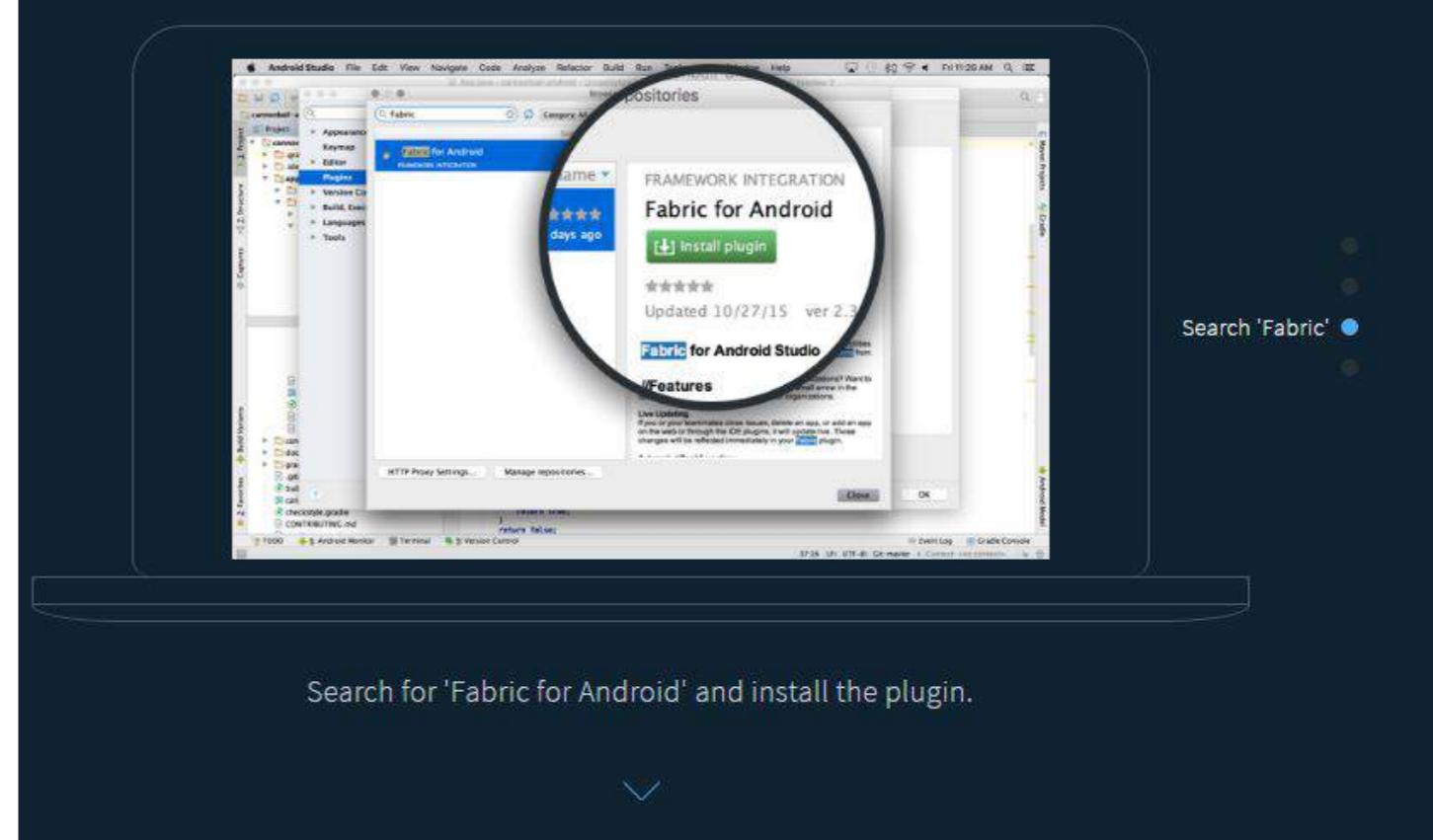
Search for 'Fabric for Android' and install the plugin.



安装插件后，重启 Android Studio，并使用Android Studio登录您的账户。

(快捷键 > **CTRL + L**)

Android Studio / IntelliJ

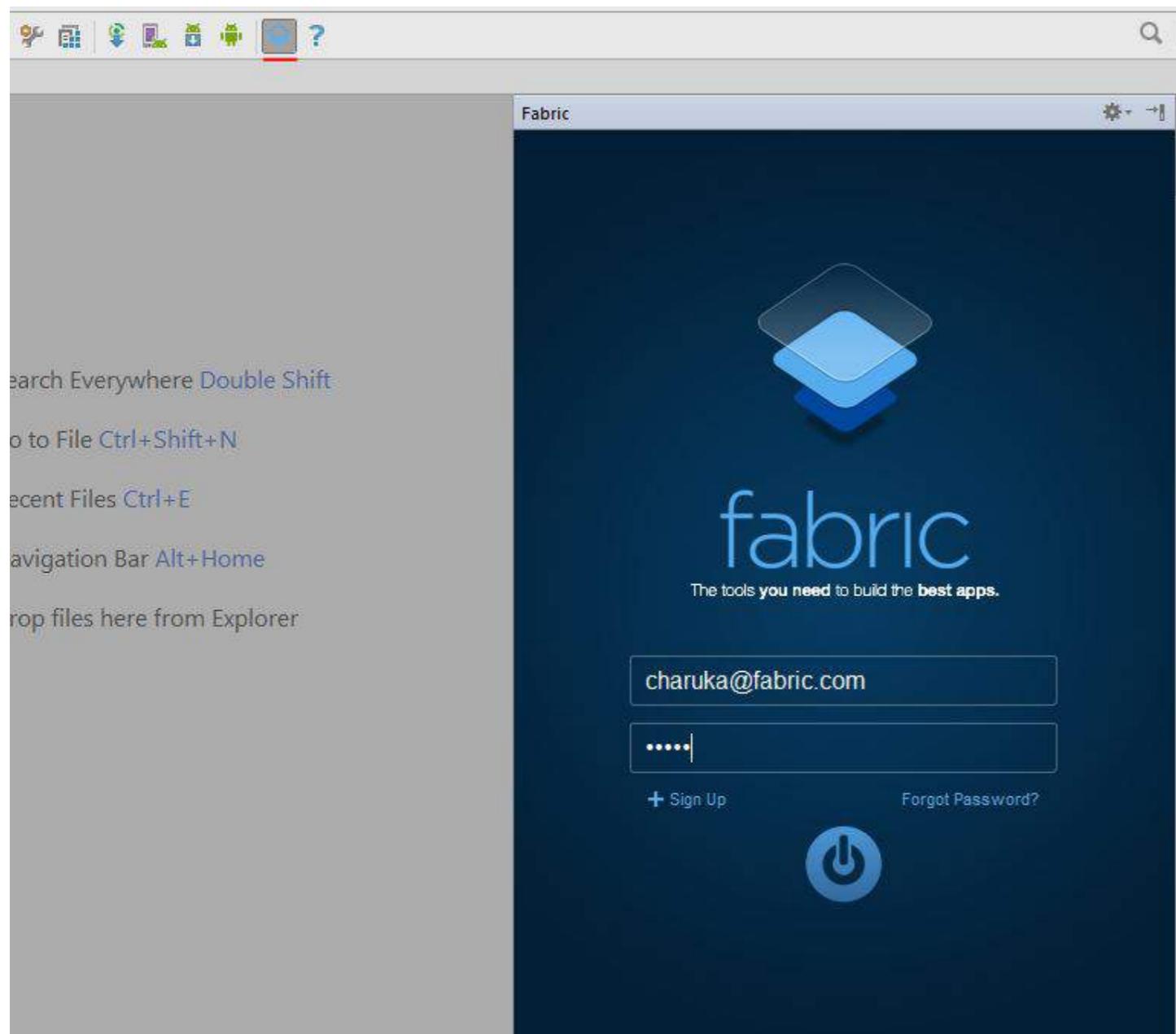


Search for 'Fabric for Android' and install the plugin.



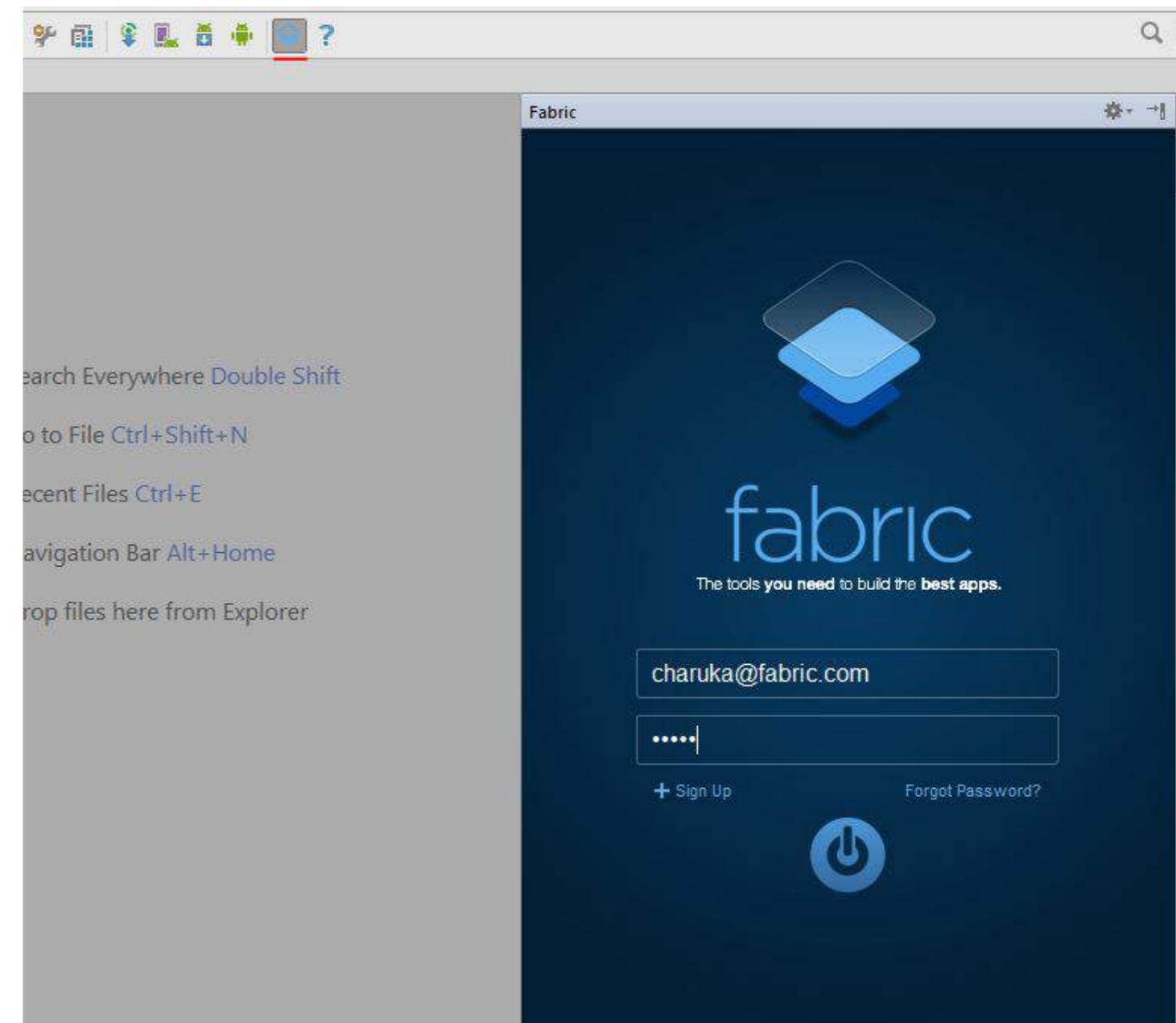
After installing the plugin, **restart** Android Studio and **login** with your account using **Android Studio**.

(short key > **CTRL + L**)



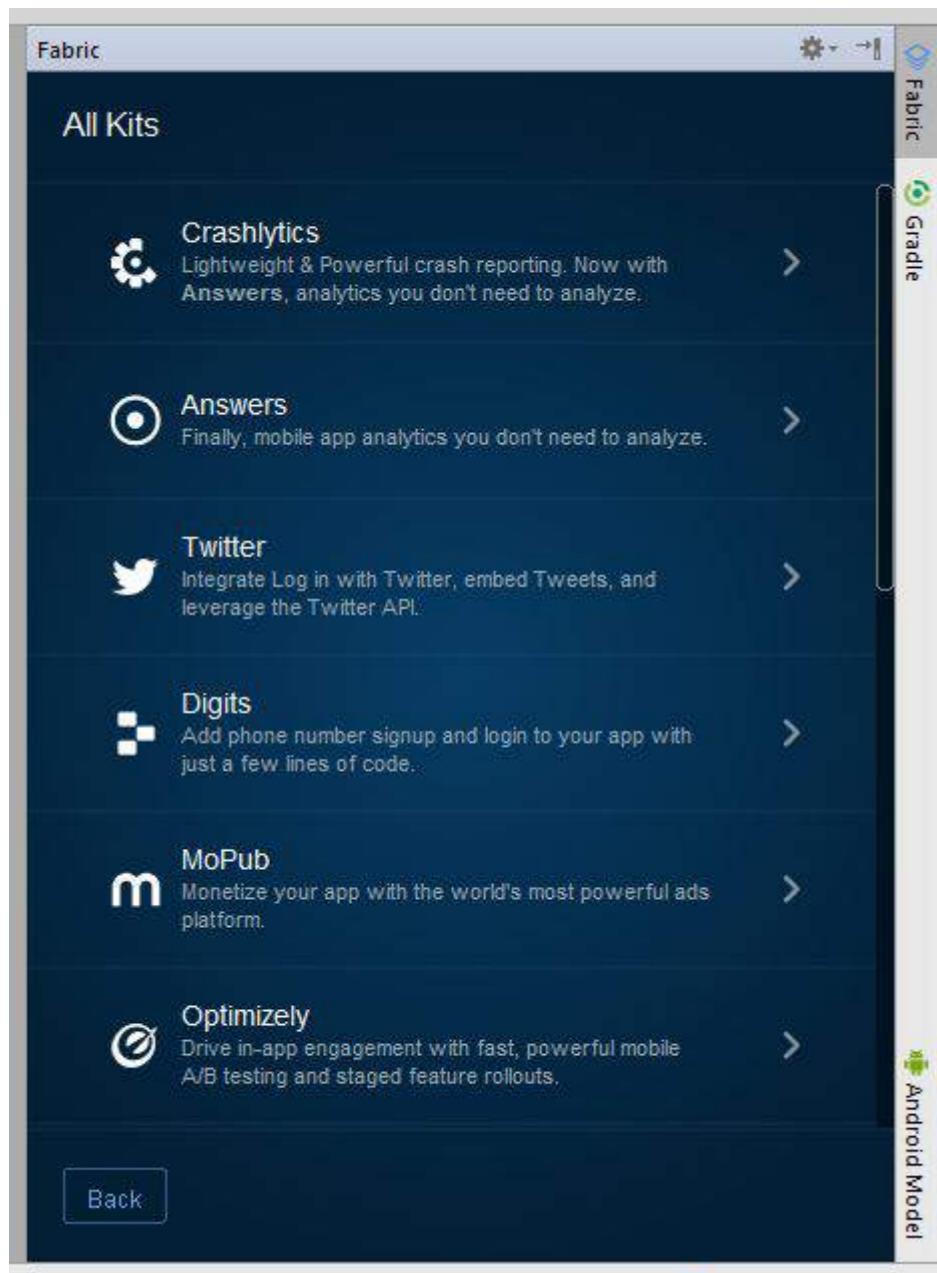
然后会显示您拥有的项目/您打开的项目，选择所需项目并点击下一步..下一步。

选择您想要添加的工具包，本例中为Crashlytics：

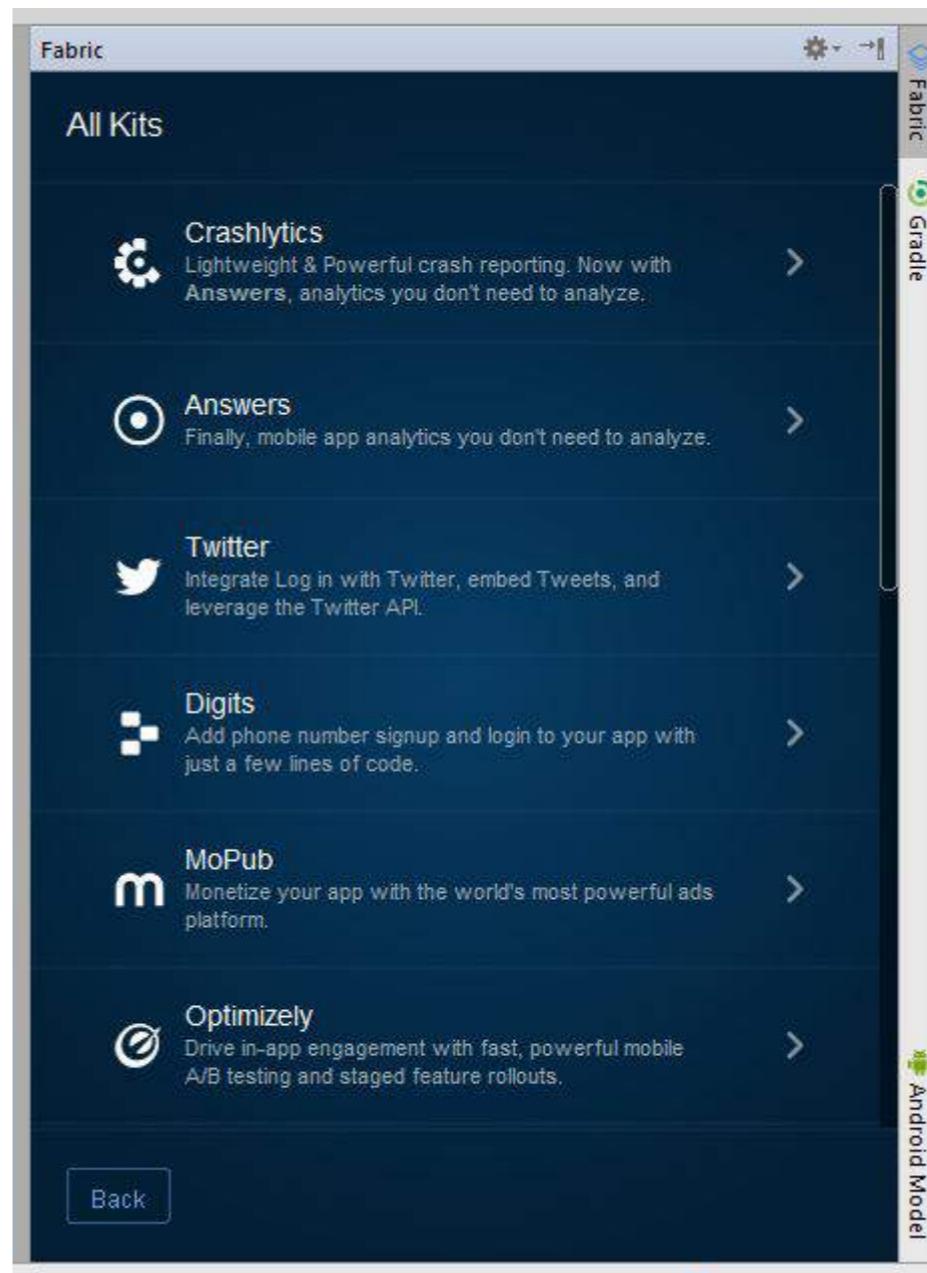


Then it will show the projects that you have / the project you opened, select the one you need and click next .. next.

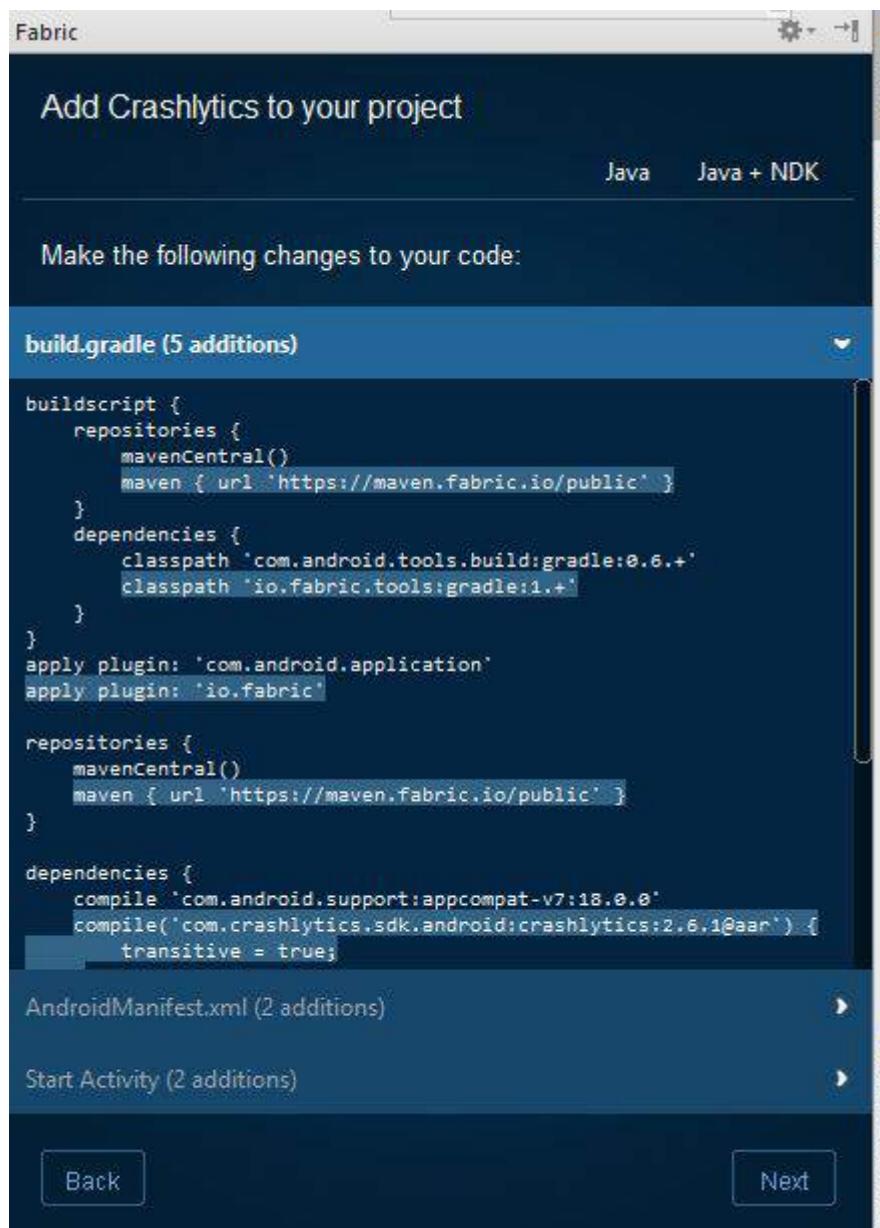
Select the kit you would like to add, for this example it is **Crashlytics**:



然后点击安装。这次你不需要像上面那样手动添加gradle插件，系统会帮你构建。



Then hit Install. You don't need to add it manually this time like above **gradle plugin**, instead it will build for you.



完成！

第125.2节：使用Sherlock捕获崩溃

Sherlock会捕获你所有的崩溃并以通知形式报告。当你点击通知时，它会打开一个活动页面，显示所有崩溃详情以及设备和应用信息。

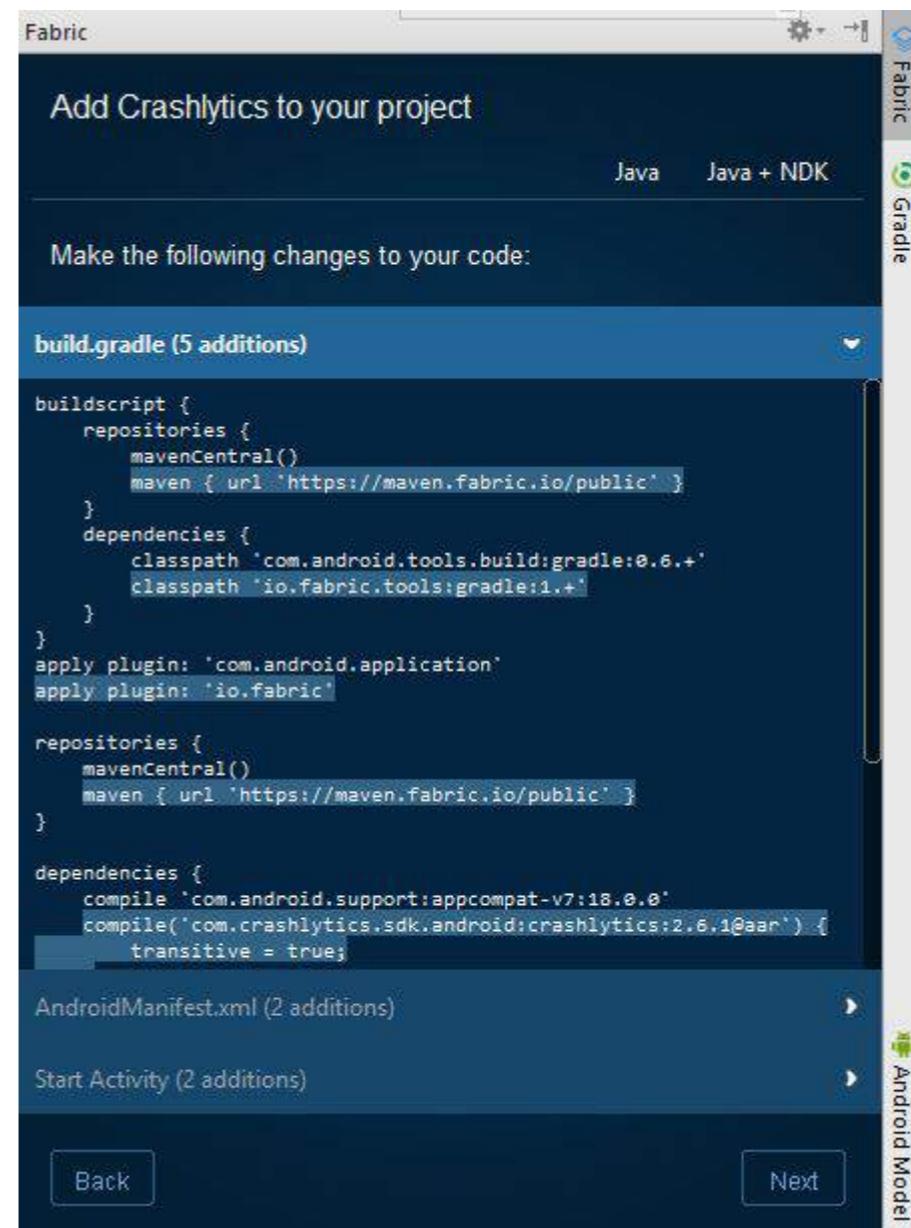
如何将Sherlock集成到你的应用中？

你只需在项目中将Sherlock作为gradle依赖添加即可。

```
dependencies {
compile('com.github.ajitsing:sherlock:1.0.1@aar') {
    transitive = true
}
}
```

同步Android Studio后，在你的Application类中初始化Sherlock。

```
package com.singhajit.login;
```



Done!

Section 125.2: Capture crashes using Sherlock

[Sherlock](#) captures all your crashes and reports them as a notification. When you tap on the notification, it opens up an activity with all the crash details along with Device and Application info

How to integrate Sherlock with your application?

You just need to add Sherlock as a gradle dependency in your project.

```
dependencies {
compile('com.github.ajitsing:sherlock:1.0.1@aar') {
    transitive = true
}
}
```

After syncing your android studio, initialize Sherlock in your Application class.

```
package com.singhajit.login;
```

```

import android.app.Application;

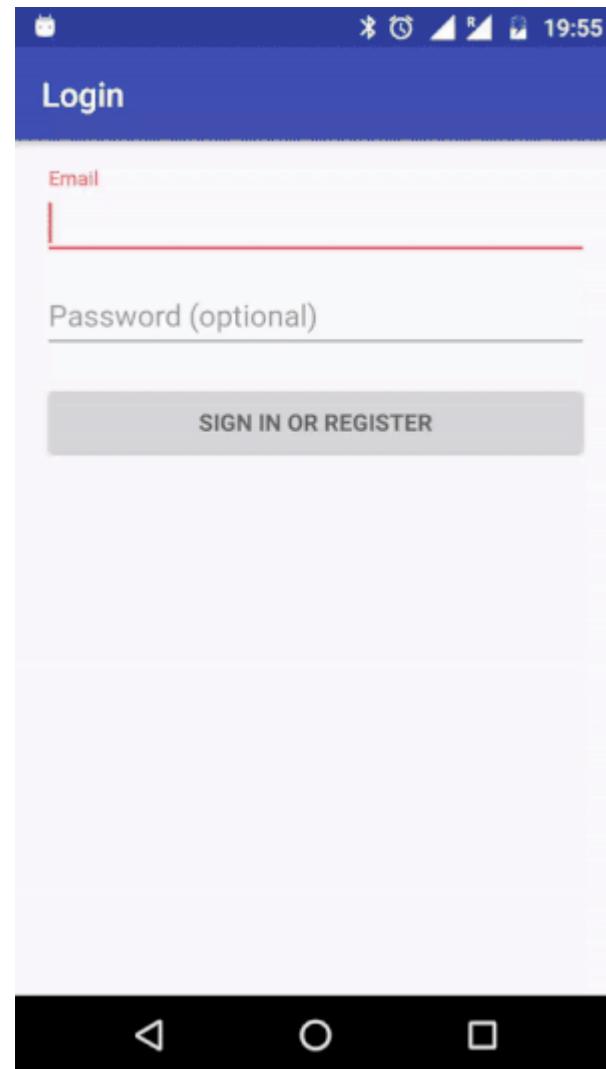
import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Sherlock.init(this);
    }
}

```

这就是你需要做的全部。此外，Sherlock 不仅仅是报告崩溃。要查看它的所有功能，请查看这篇文章。

演示



第125.3节：使用 Fabric 强制测试崩溃

添加一个按钮，点击即可触发崩溃。将此代码粘贴到你希望按钮出现的布局中。

```

<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="强制崩溃！"
    android:onClick="forceCrash"
    android:layout_centerVertical="true"/>

```

```

import android.app.Application;

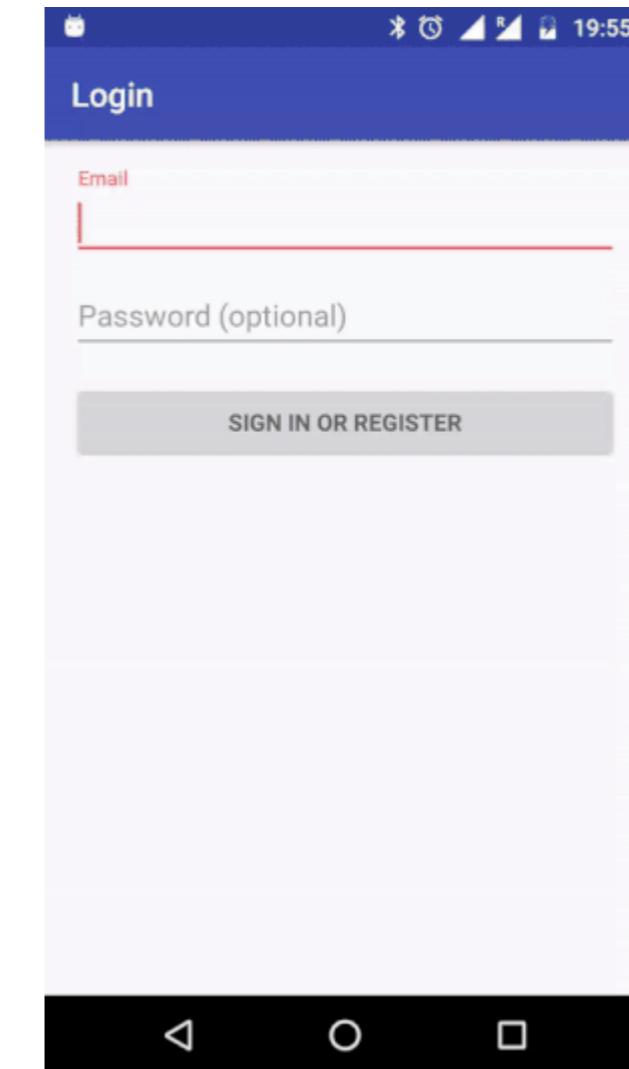
import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Sherlock.init(this);
    }
}

```

That's all you need to do. Also Sherlock does much more than just reporting a crash. To checkout all its features take a look at this [article](#).

Demo



Section 125.3: Force a Test Crash With Fabric

Add a button you can tap to trigger a crash. Paste this code into your layout where you'd like the button to appear.

```

<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="Force Crash!"
    android:onClick="forceCrash"
    android:layout_centerVertical="true"/>

```

```
        android:layout_centerHorizontal="true" />
```

抛出运行时异常

```
public void forceCrash(View view) {  
    throw new RuntimeException("这是一个崩溃");  
}
```

运行你的应用并点击新按钮以触发崩溃。一两分钟后，你应该能在Crashlytics仪表盘上看到崩溃报告，同时你也会收到一封邮件。

第125.4节：使用ACRA进行崩溃报告

步骤1：将最新的ACRA AAR依赖添加到你的应用gradle(build.gradle)中。

步骤2：在你的应用类（继承自Application的类；如果没有则创建一个）中添加@ReportsCrashes注解，并重写attachBaseContext()方法。

步骤3：在你的应用类中初始化ACRA类

```
@ReportsCrashes(  
formUri = "你选择的后端",  
    reportType = REPORT_TYPES(JSON/FORM),  
    httpMethod = HTTP_METHOD(POST/PUT),  
    formUriBasicAuthLogin = "AUTH_USERNAME",  
    formUriBasicAuthPassword = "AUTH_PASSWORD",  
    customReportContent = {  
ReportField.USER_APP_START_DATE,  
    ReportField.USER_CRASH_DATE,  
    ReportField.APP_VERSION_CODE,  
    ReportField.APP_VERSION_NAME,  
    ReportField.ANDROID_VERSION,  
    ReportField.DEVICE_ID,  
ReportField.BUILD,  
    ReportField.BRAND,  
    ReportField.DEVICE_FEATURES,  
    ReportField.PACKAGE_NAME,  
    ReportField.REPORT_ID,  
    ReportField.STACK_TRACE,  
},  
mode = NOTIFICATION_TYPE(TOAST,DIALOG,NOTIFICATION)  
resToastText = R.string.crash_text_toast)
```

```
public class MyApplication extends Application {  
    @Override  
protected void attachBaseContext(Context base) {  
    super.attachBaseContext(base);  
// Initialization of ACRA  
    ACRA.init(this);  
}
```

其中 AUTH_USERNAME 和 AUTH_PASSWORD 是您所需后端的凭据。

步骤4：在 AndroidManifest.xml 中定义 Application 类

```
<application  
    android:name=".MyApplication">  
    <service></service>
```

```
        android:layout_centerHorizontal="true" />
```

Throw a RuntimeException

```
public void forceCrash(View view) {  
    throw new RuntimeException("This is a crash");  
}
```

Run your app and tap the new button to cause a crash. In a minute or two you should be able to see the crash on your Crashlytics dashboard as well as you will get a mail.

Section 125.4: Crash Reporting with ACRA

Step 1: Add the dependency of latest [ACRA](#) AAR to your application gradle(build.gradle).

Step 2: In your application class(the class which extends Application; if not create it) Add a @ReportsCrashes annotation and override the attachBaseContext() method.

Step 3: Initialize the ACRA class in your application class

```
@ReportsCrashes(  
    formUri = "Your choice of backend",  
    reportType = REPORT_TYPES(JSON/FORM),  
    httpMethod = HTTP_METHOD(POST/PUT),  
    formUriBasicAuthLogin = "AUTH_USERNAME",  
    formUriBasicAuthPassword = "AUTH_PASSWORD",  
    customReportContent = {  
        ReportField.USER_APP_START_DATE,  
        ReportField.USER_CRASH_DATE,  
        ReportField.APP_VERSION_CODE,  
        ReportField.APP_VERSION_NAME,  
        ReportField.ANDROID_VERSION,  
        ReportField.DEVICE_ID,  
        ReportField.BUILD,  
        ReportField.BRAND,  
        ReportField.DEVICE_FEATURES,  
        ReportField.PACKAGE_NAME,  
        ReportField.REPORT_ID,  
        ReportField.STACK_TRACE,  
    },  
    mode = NOTIFICATION_TYPE(TOAST,DIALOG,NOTIFICATION)  
    resToastText = R.string.crash_text_toast)
```

```
public class MyApplication extends Application {  
    @Override  
protected void attachBaseContext(Context base) {  
    super.attachBaseContext(base);  
// Initialization of ACRA  
    ACRA.init(this);  
}
```

Where AUTH_USERNAME and AUTH_PASSWORD are the credentials of your desired [backends](#).

Step 4: Define the Application class in AndroidManifest.xml

```
<application  
    android:name=".MyApplication">  
    <service></service>
```

```
<activity></activity>
<receiver></receiver>
</application>
```

步骤5：确保您拥有internet权限以接收崩溃应用的报告

```
<uses-permission android:name="android.permission.INTERNET"/>
```

如果您想将静默报告发送到后端，只需使用以下方法实现。

```
ACRA.getErrorReporter().handleSilentException(e);
```

```
<activity></activity>
<receiver></receiver>
</application>
```

Step 5: Make sure you have internet permission to receive the report from crashed application

```
<uses-permission android:name="android.permission.INTERNET"/>
```

In case if you want to send the silent report to the backend then just use the below method to achieve it.

```
ACRA.getErrorReporter().handleSilentException(e);
```

第126章：检查互联网连接

参数	详情
上下文	Activity上下文的引用

此方法用于检查Wi-Fi是否已连接。

第126.1节：检查设备是否有互联网连接

在应用程序清单文件中添加所需的网络权限：

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

/**
 * 如果网络连接可用，将返回true
 *
 * @param context 当前上下文
 * @return boolean 如果网络连接可用则返回true
 */
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager connectivity = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivity == null) {
        Log.d("NetworkCheck", "isNetworkAvailable: No");
        return false;
    }

    // 获取所有数据接口（例如WiFi、3G、LTE等）的网络信息
    NetworkInfo[] info = connectivity.getAllNetworkInfo();

    // 确保至少有一个接口可供测试
    if (info != null) {
        // 遍历接口
        for (int i = 0; i < info.length; i++) {
            // 检查该接口是否处于连接状态
            if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                Log.d("NetworkCheck", "isNetworkAvailable: Yes");
                return true;
            }
        }
    }
    return false;
}
```

第126.2节：如何在安卓中检查网络强度？

```
ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo Info = cm.getActiveNetworkInfo();
if (Info == null || !Info.isConnectedOrConnecting()) {
    Log.i(TAG, "No connection");
} else {
    int netType = Info.getType();
    int netSubtype = Info.getSubtype();

    if (netType == ConnectivityManager.TYPE_WIFI) {
        Log.i(TAG, "Wifi connection");
    }
}
```

Chapter 126: Check Internet Connectivity

Parameter	Detail
Context	A reference of Activity context

This method is used to check whether Wi-Fi is connected or not.

Section 126.1: Check if device has internet connectivity

Add the required network permissions to the application manifest file:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

/**
 * If network connectivity is available, will return true
 *
 * @param context the current context
 * @return boolean true if a network connection is available
 */
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager connectivity = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivity == null) {
        Log.d("NetworkCheck", "isNetworkAvailable: No");
        return false;
    }

    // get network info for all of the data interfaces (e.g. WiFi, 3G, LTE, etc.)
    NetworkInfo[] info = connectivity.getAllNetworkInfo();

    // make sure that there is at least one interface to test against
    if (info != null) {
        // iterate through the interfaces
        for (int i = 0; i < info.length; i++) {
            // check this interface for a connected state
            if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                Log.d("NetworkCheck", "isNetworkAvailable: Yes");
                return true;
            }
        }
    }
    return false;
}
```

Section 126.2: How to check network strength in android?

```
ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo Info = cm.getActiveNetworkInfo();
if (Info == null || !Info.isConnectedOrConnecting()) {
    Log.i(TAG, "No connection");
} else {
    int netType = Info.getType();
    int netSubtype = Info.getSubtype();

    if (netType == ConnectivityManager.TYPE_WIFI) {
        Log.i(TAG, "Wifi connection");
    }
}
```

```

WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
List<ScanResult> scanResult = wifiManager.getScanResults();
for (int i = 0; i < scanResult.size(); i++) {
Log.d("scanResult", "Wifi 信号强度"+scanResult.get(i).level);// 信号的分贝级别
}

// 需要获取 wifi 强度
} else if (netType == ConnectivityManager.TYPE_MOBILE) {
Log.i(TAG, "GPRS/3G 连接");
// 需要区分 3G/GPRS
}
}

```

第126.3节：如何检查网络强度

要检查精确的分贝强度，请使用以下方法-

```

ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo Info = cm.getActiveNetworkInfo();
if (Info == null || !Info.isConnectedOrConnecting()) {
Log.i(TAG, "无连接");
} else {
int netType = Info.getType();
int netSubtype = Info.getSubtype();

if (netType == ConnectivityManager.TYPE_WIFI) {
Log.i(TAG, "Wifi 连接");
WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
List<ScanResult> scanResult = wifiManager.getScanResults();
for (int i = 0; i < scanResult.size(); i++) {
Log.d("scanResult", "WiFi速度"+scanResult.get(i).level);//信号的分贝级别
}

// 需要获取 wifi 强度
} else if (netType == ConnectivityManager.TYPE_MOBILE) {
Log.i(TAG, "GPRS/3G 连接");
// 需要区分 3G/GPRS
}
}

```

要检查网络类型，请使用此类-

```

public class Connectivity {
/*
* 这些常量在我的 API 级别 (7) 中尚不可用，但如果在较新版本中出现，我需要
* 处理这些情况
*/
public static final int NETWORK_TYPE_EHRPD = 14; // 级别 11
public static final int NETWORK_TYPE_EVDO_B = 12; // 级别 9
public static final int NETWORK_TYPE_HSPAP = 15; // 级别 13
public static final int NETWORK_TYPE_IDEN = 11; // 级别 8
public static final int NETWORK_TYPE_LTE = 13; // 级别 11
*/

```

```

WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
List<ScanResult> scanResult = wifiManager.getScanResults();
for (int i = 0; i < scanResult.size(); i++) {
Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db level of
signal
}

// Need to get wifi strength
} else if (netType == ConnectivityManager.TYPE_MOBILE) {
Log.i(TAG, "GPRS/3G connection");
// Need to get differentiate between 3G/GPRS
}
}

```

Section 126.3: How to check network strength

To check exact strength in decibels use this-

```

ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo Info = cm.getActiveNetworkInfo();
if (Info == null || !Info.isConnectedOrConnecting()) {
Log.i(TAG, "No connection");
} else {
int netType = Info.getType();
int netSubtype = Info.getSubtype();

if (netType == ConnectivityManager.TYPE_WIFI) {
Log.i(TAG, "Wifi connection");
WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
List<ScanResult> scanResult = wifiManager.getScanResults();
for (int i = 0; i < scanResult.size(); i++) {
Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db level of
signal
}

// Need to get wifi strength
} else if (netType == ConnectivityManager.TYPE_MOBILE) {
Log.i(TAG, "GPRS/3G connection");
// Need to get differentiate between 3G/GPRS
}
}

```

To check Network type use this Class-

```

public class Connectivity {
/*
* These constants aren't yet available in my API level (7), but I need to
* handle these cases if they come up, on newer versions
*/
public static final int NETWORK_TYPE_EHRPD = 14; // Level 11
public static final int NETWORK_TYPE_EVDO_B = 12; // Level 9
public static final int NETWORK_TYPE_HSPAP = 15; // Level 13
public static final int NETWORK_TYPE_IDEN = 11; // Level 8
public static final int NETWORK_TYPE_LTE = 13; // Level 11
*/

```

```

* 检查是否有任何连接
*
* @param context
* @return
*/
public static boolean isConnected(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();
    return (info != null && info.isConnected());
}

/**
* 检查是否有快速连接
*
* @param context
* @return
*/
public static String isConnectedFast(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();

    if ((info != null && info.isConnected())) {
        return Connectivity.isConnectionFast(info.getType(),
            info.getSubtype());
    } else
        return "No Network Access";
}

/**
* 检查连接是否快速
*
* @param type
* @param subType
* @return
*/
public static String isConnectionFast(int type, int subType) {
    if (type == ConnectivityManager.TYPE_WIFI) {
        System.out.println("CONNECTED VIA WIFI");
        return "CONNECTED VIA WIFI";
    } else if (type == ConnectivityManager.TYPE_MOBILE) {
        switch (subType) {
        case TelephonyManager.NETWORK_TYPE_1xRTT:
            return "网络类型 1xRTT"; // ~ 50-100 kbps
        case TelephonyManager.NETWORK_TYPE_CDMA:
            return "网络类型 CDMA (3G) 速度 : 2 Mbps"; // ~ 14-64 kbps
        case TelephonyManager.NETWORK_TYPE_EDGE:
            return "网络类型 EDGE (2.75G) 速度 : 100-120 Kbps"; // ~
                // 50-100
                // kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_0:
            return "网络类型 EVDO_0"; // ~ 400-1000 kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_A:
            return "网络类型 EVDO_A"; // ~ 600-1400 kbps
        case TelephonyManager.NETWORK_TYPE_GPRS:
            return "网络类型 GPRS (2.5G) 速度 : 40-50 Kbps"; // ~ 100
                // kbps
        case TelephonyManager.NETWORK_TYPE_HSDPA:
            return "网络类型 HSDPA (4G) 速度 : 2-14 Mbps"; // ~ 2-14
        }
    }
}

```

```

* Check if there is any connectivity
*
* @param context
* @return
*/
public static boolean isConnected(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();
    return (info != null && info.isConnected());
}

/**
* Check if there is fast connectivity
*
* @param context
* @return
*/
public static String isConnectedFast(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context
        .getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();

    if ((info != null && info.isConnected())) {
        return Connectivity.isConnectionFast(info.getType(),
            info.getSubtype());
    } else
        return "No Network Access";
}

/**
* Check if the connection is fast
*
* @param type
* @param subType
* @return
*/
public static String isConnectionFast(int type, int subType) {
    if (type == ConnectivityManager.TYPE_WIFI) {
        System.out.println("CONNECTED VIA WIFI");
        return "CONNECTED VIA WIFI";
    } else if (type == ConnectivityManager.TYPE_MOBILE) {
        switch (subType) {
        case TelephonyManager.NETWORK_TYPE_1xRTT:
            return "NETWORK TYPE 1xRTT"; // ~ 50-100 kbps
        case TelephonyManager.NETWORK_TYPE_CDMA:
            return "NETWORK TYPE CDMA (3G) Speed: 2 Mbps"; // ~ 14-64 kbps
        case TelephonyManager.NETWORK_TYPE_EDGE:
            return "NETWORK TYPE EDGE (2.75G) Speed: 100-120 Kbps"; // ~
                // 50-100
                // kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_0:
            return "NETWORK TYPE EVDO_0"; // ~ 400-1000 kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_A:
            return "NETWORK TYPE EVDO_A"; // ~ 600-1400 kbps
        case TelephonyManager.NETWORK_TYPE_GPRS:
            return "NETWORK TYPE GPRS (2.5G) Speed: 40-50 Kbps"; // ~ 100
                // kbps
        case TelephonyManager.NETWORK_TYPE_HSDPA:
            return "NETWORK TYPE HSDPA (4G) Speed: 2-14 Mbps"; // ~ 2-14
        }
    }
}

```

```
// Mbps
case TelephonyManager.NETWORK_TYPE_HSPA:
    return "网络类型 HSPA (4G) 速度 : 0.7-1.7 Mbps"; // ~
                                                // 700-1700
                                                // kbps
case TelephonyManager.NETWORK_TYPE_HSUPA:
    return "网络类型 HSUPA (3G) 速度 : 1-23 Mbps"; // ~ 1-23
                                                // Mbps
case TelephonyManager.NETWORK_TYPE_UMTS:
    return "网络类型 UMTS (3G) 速度 : 0.4-7 Mbps"; // ~ 400-7000
                                                // kbps
// API 级别 7 中尚不可用
case Connectivity.NETWORK_TYPE_EHRPD:
    return "网络类型 EHRPD"; // ~ 1-2 Mbps
case Connectivity.NETWORK_TYPE_EVDO_B:
    return "网络类型 EVDO_B"; // ~ 5 Mbps
case Connectivity.NETWORK_TYPE_HSPAP:
    return "网络类型 HSPA+ (4G) 速度 : 10-20 Mbps"; // ~ 10-20
                                                // Mbps
case Connectivity.NETWORK_TYPE_IDEN:
    return "网络类型识别"; // ~25 kbps
case Connectivity.NETWORK_TYPE_LTE:
    return "网络类型 LTE (4G) 速度 : 10+ Mbps"; // ~ 10+ Mbps
    // 未知
case TelephonyManager.NETWORK_TYPE_UNKNOWN:
    return "网络类型 未知";
default:
    return "";
}
} else {
    return "";
}
}
```

```
// Mbps
case TelephonyManager.NETWORK_TYPE_HSPA:
    return "NETWORK TYPE HSPA (4G) Speed: 0.7-1.7 Mbps"; // ~
                                                               // 700-1700
                                                               // kbps
case TelephonyManager.NETWORK_TYPE_HSUPA:
    return "NETWORK TYPE HSUPA (3G) Speed: 1-23 Mbps"; // ~ 1-23
                                                               // Mbps
case TelephonyManager.NETWORK_TYPE_UMTS:
    return "NETWORK TYPE UMTS (3G) Speed: 0.4-7 Mbps"; // ~ 400-7000
                                                               // kbps
// NOT AVAILABLE YET IN API LEVEL 7
case Connectivity.NETWORK_TYPE_EHRPD:
    return "NETWORK TYPE EHRPD"; // ~ 1-2 Mbps
case Connectivity.NETWORK_TYPE_EVDO_B:
    return "NETWORK_TYPE_EVDO_B"; // ~ 5 Mbps
case Connectivity.NETWORK_TYPE_HSPAP:
    return "NETWORK TYPE HSPA+ (4G) Speed: 10-20 Mbps"; // ~ 10-20
                                                               // Mbps
case Connectivity.NETWORK_TYPE_IDEN:
    return "NETWORK TYPE IDEN"; // ~25 kbps
case Connectivity.NETWORK_TYPE_LTE:
    return "NETWORK TYPE LTE (4G) Speed: 10+ Mbps"; // ~ 10+ Mbps
                                                               // Unknown
case TelephonyManager.NETWORK_TYPE_UNKNOWN:
    return "NETWORK TYPE UNKNOWN";
default:
    return "";
}
} else {
    return "";
}
}
```

第127章：为安卓应用创建自己的库

第127.1节：创建一个可在Jitpack.io上使用的库

执行以下步骤以创建库：

1. 创建一个GitHub账户。
2. 创建一个包含你的库项目的Git仓库。
3. 通过添加以下代码修改你的库项目的build.gradle文件：

```
apply plugin: 'com.github.dcendents.android-maven'

...
// 构建包含源文件的jar包。
task sourcesJar(type: Jar) {
    from android.sourceSets.main.java.srcDirs
    classifier = 'sources'
}

task javadoc(type: Javadoc) {
    failOnError false
    source = android.sourceSets.main.java.sourceFiles
    classpath += project.files(android.getBootClasspath().join(File.pathSeparator))
    classpath += configurations.compile
}

// 构建包含javadoc的jar包。
task javadocJar(type: Jar, dependsOn: javadoc) {
    classifier = 'javadoc'
    from javadoc.destinationDir
}

artifacts {
    archives sourcesJar
    archives javadocJar
}
```

确保你已将上述更改提交/推送到GitHub。

4. 从GitHub上当前的代码创建一个发布版本。
5. 在你的代码上运行gradlew install。
6. 你的库现在可以通过以下依赖项使用：

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

第127.2节：创建库项目

要创建一个库，你应该使用[文件->新建->新建模块->Android库](#)。这样会创建一个基础的库项目。

Chapter 127: Creating your own libraries for Android applications

Section 127.1: Create a library available on Jitpack.io

Perform the following steps to create the library:

1. Create a GitHub account.
2. Create a Git repository containing your library project.
3. Modify your library project's build.gradle file by adding the following code:

```
apply plugin: 'com.github.dcendents.android-maven'

...
// Build a jar with source files.
task sourcesJar(type: Jar) {
    from android.sourceSets.main.java.srcDirs
    classifier = 'sources'
}

task javadoc(type: Javadoc) {
    failOnError false
    source = android.sourceSets.main.java.sourceFiles
    classpath += project.files(android.getBootClasspath().join(File.pathSeparator))
    classpath += configurations.compile
}

// Build a jar with javadoc.
task javadocJar(type: Jar, dependsOn: javadoc) {
    classifier = 'javadoc'
    from javadoc.destinationDir
}

artifacts {
    archives sourcesJar
    archives javadocJar
}
```

Make sure that you commit/push the above changes to GitHub.

4. Create a release from the current code on Github.
5. Run gradlew install on your code.
6. Your library is now available by the following dependency:

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

Section 127.2: Creating library project

To create a library, you should use [File -> New -> New Module -> Android Library](#). This will create a basic library project.

完成后，你必须拥有一个按以下方式设置的项目：

```
[项目根目录]
  [库根目录]
    [gradle]
  build.gradle //项目级别
    gradle.properties
    gradlew
  gradlew.bat
  local.properties
  settings.gradle //这很重要！
```

您的 settings.gradle 文件必须包含以下内容：

```
include ':[库根目录]'
```

您的 [库根目录] 必须包含以下内容：

```
[libs]
[src]
  [main]
    [java]
      [库包]
  [测试]
    [java]
      [库包]
build.gradle //“app”级别
proguard-规则.pro
```

您的“app”级别build.gradle文件必须包含以下内容：

```
应用插件: 'com.android.library'

android {
compileSdkVersion 23
  buildToolsVersion "23.0.2"

  defaultConfig {
minSdkVersion 14
    targetSdkVersion 23
  }
}
```

这样，您的项目应该可以正常运行！

第127.3节：在项目中作为模块使用库

要使用该库，您必须通过以下行将其作为依赖项包含：

```
编译项目(':[library root directory]')
```

When that's done, you must have a project that is set up the following manner:

```
[project root directory]
  [library root directory]
    [gradle]
  build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
  settings.gradle //this is important!
```

Your settings.gradle file must contain the following:

```
include ':[library root directory]'
```

Your [library root directory] must contain the following:

```
[libs]
[src]
  [main]
    [java]
      [library package]
  [test]
    [java]
      [library package]
build.gradle //“app”-level
proguard-rules.pro
```

Your “app”-level build.gradle file must contain the following:

```
apply plugin: 'com.android.library'

android {
  compileSdkVersion 23
  buildToolsVersion "23.0.2"

  defaultConfig {
    minSdkVersion 14
    targetSdkVersion 23
  }
}
```

With that, your project should be working fine!

Section 127.3: Using library in project as a module

To use the library, you must include it as a dependency with the following line:

```
compile project(':[library root directory]')
```

第128章：设备显示指标

第128.1节：获取屏幕像素尺寸

要获取屏幕的宽度和高度（以像素为单位），我们可以利用WindowManagers的显示指标。

```
// 获取显示指标  
DisplayMetrics metrics = new DisplayMetrics();  
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

这些DisplayMetrics包含了设备屏幕的一系列信息，比如其密度或尺寸：

```
// 获取像素宽度和高度  
Integer heightPixels = metrics.heightPixels;  
Integer widthPixels = metrics.widthPixels;
```

第128.2节：获取屏幕密度

要获取屏幕密度，我们也可以使用Windowmanagers DisplayMetrics。以下是一个快速示例：

```
// 获取dpi密度  
DisplayMetrics metrics = new DisplayMetrics();  
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);  
int densityInDpi = metrics.densityDpi;
```

第128.3节：px转dp， dp转px的公式转换

DP转像素：

```
private int dpToPx(int dp)  
{  
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);  
}
```

像素转DP：

```
private int pxToDp(int px)  
{  
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);  
}
```

Chapter 128: Device Display Metrics

Section 128.1: Get the screens pixel dimensions

To retrieve the screens width and height in pixels, we can make use of the [WindowManagers](#) display metrics.

```
// Get display metrics  
DisplayMetrics metrics = new DisplayMetrics();  
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

These [DisplayMetrics](#) hold a series of information about the devices screen, like its density or size:

```
// Get width and height in pixel  
Integer heightPixels = metrics.heightPixels;  
Integer widthPixels = metrics.widthPixels;
```

Section 128.2: Get screen density

To get the screens density, we also can make use of the [Windowmanagers DisplayMetrics](#). This is a quick example:

```
// Get density in dpi  
DisplayMetrics metrics = new DisplayMetrics();  
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);  
int densityInDpi = metrics.densityDpi;
```

Section 128.3: Formula px to dp, dp to px conversation

DP to Pixel:

```
private int dpToPx(int dp)  
{  
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);  
}
```

Pixel to DP:

```
private int pxToDp(int px)  
{  
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);  
}
```

第129章：构建向后兼容的应用程序

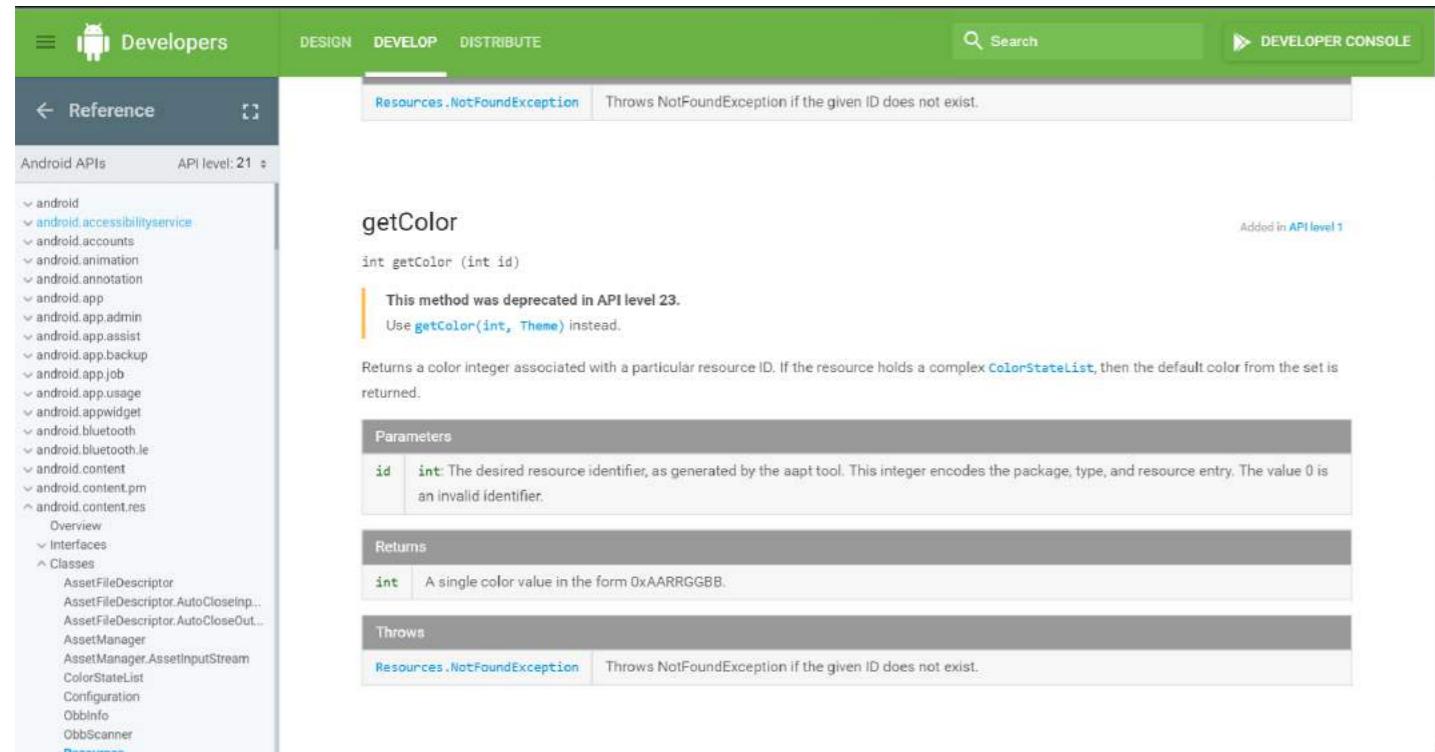
第129.1节：如何处理已弃用的API

开发过程中，开发者几乎不可能不遇到已弃用的API。已弃用的程序元素是指程序员被建议不要使用的元素，通常是因为它存在安全隐患，或者有更好的替代方案。当在非弃用代码中使用或重写已弃用的程序元素时，编译器和分析工具（如LINT）会发出警告。

在Android Studio中，已弃用的API通常会以删除线的形式标识。下面的示例中，方法`.getColor(int id)`已被弃用：

```
getResources().getColor(R.color.colorAccent);
```

如果可能，建议开发者使用替代的API和元素。可以通过访问该库的Android文档并查看“Added in API level x”部分，来检查库的向后兼容性：



The screenshot shows the Android Developers API reference for the `getColor` method. The method is marked as deprecated in API level 23, with a note to use `getColor(int, Theme)` instead. The documentation also states that it returns a color integer associated with a particular resource ID.

如果你需要使用的API与用户使用的Android版本不兼容，应在使用该库之前检查用户的API级别。例如：

```
//检查运行设备的API级别
if (Build.VERSION.SDK_INT < 23) {
    //用于兼容23以下的API级别
    int color = getResources().getColor(R.color.colorPrimary);
} else {
    int color = getResources().getColor(R.color.colorPrimary, getActivity().getTheme());
}
```

使用此方法可确保您的应用既兼容新的Android版本，也兼容现有版本。

Chapter 129: Building Backwards Compatible Apps

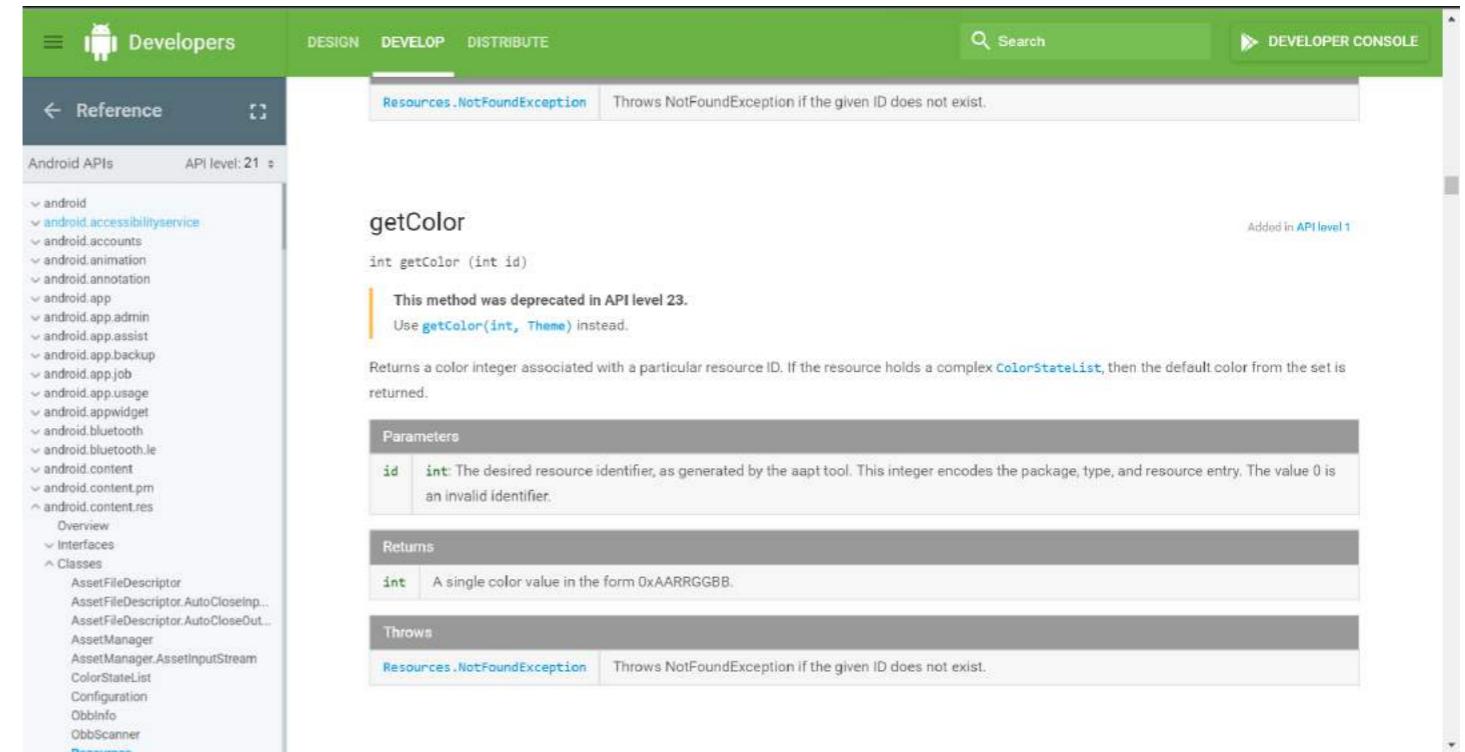
Section 129.1: How to handle deprecated API

It is unlikely for a developer to not come across a deprecated API during a development process. A deprecated program element is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists. Compilers and analyzers (like LINT) warn when a deprecated program element is used or overridden in non-deprecated code.

A deprecated API is usually identified in Android Studio using a strikeout. In the example below, the method `.getColor(int id)` is deprecated:

```
getResources().getColor(R.color.colorAccent));
```

If possible, developers are encouraged to use alternative APIs and elements. It is possible to check backwards compatibility of a library by visiting the Android documentation for the library and checking the "Added in API level x" section:



The screenshot shows the Android Developers API reference for the `getColor` method. The method is marked as deprecated in API level 23, with a note to use `getColor(int, Theme)` instead. The documentation also states that it returns a color integer associated with a particular resource ID.

In the case that the API you need to use is not compatible with the Android version that your users are using, you should check for the API level of the user before using that library. For example:

```
//Checks the API level of the running device
if (Build.VERSION.SDK_INT < 23) {
    //use for backwards compatibility with API levels below 23
    int color = getResources().getColor(R.color.colorPrimary);
} else {
    int color = getResources().getColor(R.color.colorPrimary, getActivity().getTheme());
}
```

Using this method ensures that your app will remain compatible with new Android versions as well as existing versions.

更简单的替代方案：使用支持库

如果使用支持库，通常有静态辅助方法可以用更少的客户端代码完成相同任务。替代上面的if/else代码块，只需使用：

```
final int color = android.support.v4.content.ContextCompat  
    .getColor(context, R.color.colorPrimary);
```

大多数已弃用的方法都有新的签名和许多新功能，这些功能可能无法在旧版本上使用，但都有类似的兼容性辅助方法。
要查找其他方法，可以浏览支持库中的类，如ContextCompat、ViewCompat等。

Easier alternative: Use the Support Library

If the Support Libraries are used, often there are static helper methods to accomplish the same task with less client code. Instead of the if/else block above, just use:

```
final int color = android.support.v4.content.ContextCompat  
    .getColor(context, R.color.colorPrimary);
```

Most deprecated methods that have newer methods with a different signature and many new features that may not have been able to be used on older versions have compatibility helper methods like this. To find others, browse through the support library for classes like ContextCompat, ViewCompat, etc.

第130章：加载器

类	描述
加载器管理器	一个与Activity或Fragment关联的抽象类，用于管理一个或多个加载器实例。
LoaderManager.LoaderCallbacks	一个回调接口，供客户端与加载器管理器交互。
加载器	执行异步数据加载的抽象类。
异步任务加载器	提供AsyncTask来执行工作的抽象加载器。
游标加载器	AsyncTaskLoader 的一个子类，用于查询 ContentResolver 并返回一个 Cursor。

如果你想在 onCreate 方法被调用时在后台加载数据，Loader 是防止内存泄漏的好选择。例如，当我们在 onCreate 方法中执行 AsyncTask 并旋转屏幕时，活动会被重新创建，这将再次执行另一个 AsyncTask，因此可能会有两个 AsyncTask 并行运行，而不像 Loader 那样会继续执行之前启动的后台进程。

第130.1节：基本的 AsyncTaskLoader

AsyncTaskLoader是一个抽象的Loader，提供了一个AsyncTask来执行工作。

以下是一些基本实现：

```
final class BasicLoader extends AsyncTaskLoader<String> {

    public BasicLoader(Context context) {
        super(context);
    }

    @Override
    public String loadInBackground() {
        // 一些工作，例如从互联网加载内容
        return "OK";
    }

    @Override
    public void deliverResult(String data) {
        if (isStarted()) {
            // 如果 Loader 当前已启动，则传递结果
            super.deliverResult(data);
        }
    }

    @Override
    protected void onStartLoading() {
        // 开始加载
        forceLoad();
    }

    @Override
    protected void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected void onReset() {
        super.onReset();
    }
}
```

Chapter 130: Loader

Class	Description
LoaderManager	An abstract class associated with an Activity or Fragment for managing one or more Loader instances.
LoaderManager.LoaderCallbacks	A callback interface for a client to interact with the LoaderManager.
Loader	An abstract class that performs asynchronous loading of data.
AsyncTaskLoader	Abstract loader that provides an AsyncTask to do the work.
CursorLoader	A subclass of AsyncTaskLoader that queries the ContentResolver and returns a Cursor.

Loader is good choice for prevent memory leak if you want to load data in background when oncreate method is called. For example when we execute AsyncTask in oncreate method and we rotate the screen so the activity will recreate which will execute another AsyncTask again, so probably two AsyncTask running in parallel together rather than like loader which will continue the background process we executed before.

Section 130.1: Basic AsyncTaskLoader

AsyncTaskLoader is an abstract Loader that provides an AsyncTask to do the work.

Here some basic implementation:

```
final class BasicLoader extends AsyncTaskLoader<String> {

    public BasicLoader(Context context) {
        super(context);
    }

    @Override
    public String loadInBackground() {
        // Some work, e.g. load something from internet
        return "OK";
    }

    @Override
    public void deliverResult(String data) {
        if (isStarted()) {
            // Deliver result if loader is currently started
            super.deliverResult(data);
        }
    }

    @Override
    protected void onStartLoading() {
        // Start loading
        forceLoad();
    }

    @Override
    protected void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected void onReset() {
        super.onReset();
    }
}
```

```
// 确保加载器已停止  
onStopLoading();  
}  
}
```

通常Loader在活动的onCreate()方法中初始化，或在片段的onActivityCreated()中初始化。通常活动或片段实现LoaderManager.LoaderCallbacks接口：

```
public class MainActivity extends Activity implements LoaderManager.LoaderCallbacks<String> {  
  
    // 加载器的唯一ID  
    private static final int LDR_BASIC_ID = 1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // 初始化加载器；一些数据可以作为第二个参数传递，而不是 Bundle.EMPTY  
        getLoaderManager().initLoader(LDR_BASIC_ID, Bundle.EMPTY, this);  
    }  
  
    @Override  
    public Loader<String> onCreateLoader(int id, Bundle args) {  
        return new BasicLoader(this);  
    }  
  
    @Override  
    public void onLoadFinished(Loader<String> loader, String data) {  
        Toast.makeText(this, data, Toast.LENGTH_LONG).show();  
    }  
  
    @Override  
    public void onLoaderReset(Loader<String> loader) {  
    }  
}
```

在此示例中，当加载器完成时，将显示带有结果的吐司提示。

第130.2节：带缓存的AsyncTaskLoader

缓存已加载的结果是一个良好实践，以避免多次加载相同数据。

要使缓存失效，应调用 onContentChanged()。如果加载器已经启动，将调用 forceLoad()，否则（如果加载器处于停止状态）加载器将通过 takeContentChanged()检查来理解内容变化。

备注：onContentChanged()必须从进程的主线程调用。

Javadocs 关于 takeContentChanged() 的说明：

获取当前标志，指示加载器在停止期间内容是否发生了变化。如果发生了变化，返回 true 并清除该标志。

```
public abstract class BaseLoader<T> extends AsyncTaskLoader<T> {  
  
    // 缓存的结果保存在这里  
    private final AtomicReference<T> cache = new AtomicReference<>();
```

```
// Ensure the loader is stopped  
onStopLoading();  
}
```

Typically Loader is initialized within the activity's onCreate() method, or within the fragment's onActivityCreated(). Also usually activity or fragment implements LoaderManager.LoaderCallbacks interface:

```
public class MainActivity extends Activity implements LoaderManager.LoaderCallbacks<String> {  
  
    // Unique id for loader  
    private static final int LDR_BASIC_ID = 1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // Initialize loader; Some data can be passed as second param instead of Bundle.EMPTY  
        getLoaderManager().initLoader(LDR_BASIC_ID, Bundle.EMPTY, this);  
    }  
  
    @Override  
    public Loader<String> onCreateLoader(int id, Bundle args) {  
        return new BasicLoader(this);  
    }  
  
    @Override  
    public void onLoadFinished(Loader<String> loader, String data) {  
        Toast.makeText(this, data, Toast.LENGTH_LONG).show();  
    }  
  
    @Override  
    public void onLoaderReset(Loader<String> loader) {  
    }  
}
```

In this example, when loader completed, toast with result will be shown.

Section 130.2: AsyncTaskLoader with cache

It's a good practice to cache loaded result to avoid multiple loading of same data.

To invalidate cache onContentChanged() should be called. If loader has been already started, forceLoad() will be called, otherwise (if loader in stopped state) loader will be able to understand content change with takeContentChanged() check.

Remark: onContentChanged() must be called from the process's main thread.

Javadocs says about takeContentChanged():

Take the current flag indicating whether the loader's content had changed while it was stopped. If it had, true is returned and the flag is cleared.

```
public abstract class BaseLoader<T> extends AsyncTaskLoader<T> {  
  
    // Cached result saved here  
    private final AtomicReference<T> cache = new AtomicReference<>();
```

```

public BaseLoader(@NonNull final Context context) {
    super(context);
}

@Override
public final void deliverResult(final T data) {
    if (!isReset()) {
        // 保存加载的结果
        cache.set(data);
        if (isStarted()) {
            super.deliverResult(data);
        }
    }
}

@Override
protected final void onStartLoading() {
    // 注册观察者
    registerObserver();

    final T cached = cache.get();
    // 如果后台内容发生变化
    // 或者我们从未加载过任何数据，则开始新的加载
    if (takeContentChanged() || cached == null) {
        forceLoad();
    } else {
        deliverResult(cached);
    }
}

@Override
public final void onStopLoading() {
    cancelLoad();
}

@Override
protected final void onReset() {
    super.onReset();
    onStopLoading();
    // 清除缓存并移除观察者
    cache.set(null);
    unregisterObserver();
}

/* virtual */
protected void registerObserver() {
    // 在此注册观察者，调用 onContentChanged() 以使缓存失效
}

/* virtual */
protected void unregisterObserver() {
    // 移除观察者
}

```

第130.3节：重新加载

要使旧数据失效并重新启动现有加载器，可以使用 `restartLoader()` 方法：

```

private void reload() {
    getLoaderManager().restartLoader(LOADER_ID, Bundle.EMPTY, this);
}

```

```

public BaseLoader(@NonNull final Context context) {
    super(context);
}

@Override
public final void deliverResult(final T data) {
    if (!isReset()) {
        // Save loaded result
        cache.set(data);
        if (isStarted()) {
            super.deliverResult(data);
        }
    }
}

@Override
protected final void onStartLoading() {
    // Register observers
    registerObserver();

    final T cached = cache.get();
    // Start new loading if content changed in background
    // or if we never loaded any data
    if (takeContentChanged() || cached == null) {
        forceLoad();
    } else {
        deliverResult(cached);
    }
}

@Override
public final void onStopLoading() {
    cancelLoad();
}

@Override
protected final void onReset() {
    super.onReset();
    onStopLoading();
    // Clear cache and remove observers
    cache.set(null);
    unregisterObserver();
}

/* virtual */
protected void registerObserver() {
    // Register observers here, call onContentChanged() to invalidate cache
}

/* virtual */
protected void unregisterObserver() {
    // Remove observers
}

```

Section 130.3: Reloading

To invalidate your old data and restart existing loader you can use `restartLoader()` method:

```

private void reload() {
    getLoaderManager().restartLoader(LOADER_ID, Bundle.EMPTY, this);
}

```

}

第130.4节：使用 Bundle 传递参数

您可以通过 Bundle 传递参数：

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

在 onCreateLoader 中获取值：

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

}

Section 130.4: Pass parameters using a Bundle

You can pass parameters by Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Get the value in onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

第131章：ProGuard - 混淆和压缩你的代码

第131.1节：一些常用库的规则

目前包含以下库的规则：

1. ButterKnife
2. RxJava
- 3.Android Support Library
- 4.Android Design Support Library
- 5.Retrofit
- 6.Gson 和 Jackson
7. Otto
- 8.Crashlytics
- 9.Picasso
- 10.Volley
- 11.OkHttp3
- 12.Parcelable

```
#Butterknife
-keep class butterknife.** { *; }
-keepnames class * { @butcherknife.Bind *; }

-dontwarn butterknife.internal.**
-keep class **$ViewBinder { *; }

-keepclasseswithmembernames class * {
    @butcherknife.* <fields>;
}

-keepclasseswithmembernames class * {
    @butcherknife.* <methods>;
}

# rxjava
-保持类 rx.调度器.Schedulers {
    public static <方法>;
}
    ...
        class rx.schedulers.ImmediateScheduler {
            public <methods>;
        }
        ...
        class rx.schedulers.TestScheduler {
            public <methods>;
        }
        ...
        class rx.schedulers.Schedulers {
            public static ** test();
            ...
                class rx.internal.util.unsafe.*ArrayQueue*Field* {
                    long producerIndex;
                    long consumerIndex;
                    ...
                        class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
                            long producerNode;
                            long consumerNode;
                        }
                    ...
                }
            ...
        }
    ...
}
```

Chapter 131: ProGuard - Obfuscating and Shrinking your code

Section 131.1: Rules for some of the widely used Libraries

Currently it contains rules for following libraries:

1. ButterKnife
2. RxJava
3. Android Support Library
4. Android Design Support Library
5. Retrofit
6. Gson and Jackson
7. Otto
8. Crashlytics
9. Picasso
10. Volley
11. OkHttp3
12. Parcelable

```
#Butterknife
-keep class butterknife.** { *; }
-keepnames class * { @butcherknife.Bind *; }

-dontwarn butterknife.internal.**
-keep class **$ViewBinder { *; }

-keepclasseswithmembernames class * {
    @butcherknife.* <fields>;
}

-keepclasseswithmembernames class * {
    @butcherknife.* <methods>;
}

# rxjava
-keep class rx.schedulers(Schedulers {
    public static <methods>;
}
    ...
        class rx.schedulers.ImmediateScheduler {
            public <methods>;
        }
        ...
        class rx.schedulers.TestScheduler {
            public <methods>;
        }
        ...
        class rx.schedulers.Schedulers {
            public static ** test();
            ...
                class rx.internal.util.unsafe.*ArrayQueue*Field* {
                    long producerIndex;
                    long consumerIndex;
                    ...
                        class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
                            long producerNode;
                            long consumerNode;
                        }
                    ...
                }
            ...
        }
    ...
}
```

```

# 支持库
-dontwarn android.support.**
-dontwarn android.support.v4.**
-keep class android.support.v4.** { *; }
-keep interface android.support.v4.** { *; }
-dontwarn android.support.v7.**
-keep class android.support.v7.** { *; }
-keep interface android.support.v7.** { *; }

# 支持设计
-dontwarn android.support.design.**
-keep class android.support.design.** { *; }
-keep interface android.support.design.** { *; }
-保留public class android.support.design.R$* { *; }

# retrofit
-不警告 okio.**
-保留属性 Signature
-保留属性 *Annotation*
-保留 class com.squareup.okhttp.** { *; }
-保留 interface com.squareup.okhttp.** { *; }
-不警告 com.squareup.okhttp.**

-不警告 rx.**
-不警告 retrofit.**
-保留 class retrofit.** { *; }
- keepclasseswithmembers class * {
    @retrofit.http.* <methods>;
}

- keep class sun.misc.Unsafe { *; }
#your package path where your gson models are stored
- keep class com.abc.model.** { *; }

# Keep these for GSON and Jackson
-保留属性 Signature
-保留属性 *Annotation*
- keepattributes EnclosingMethod
- keep class sun.misc.Unsafe { *; }
- keep class com.google.gson.** { *; }

#keep otto
-保留属性 *Annotation*
- keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}

# Crashlytics 2.+
- keep class com.crashlytics.** { *; }
- keep class com.crashlytics.android.**
-保留属性 SourceFile, LineNumberTable, *注解*# 如果您使用自定义异常
, 请添加此行, 以便在混淆过程中跳过自定义异常类型:

-保留public class * extends java.lang.Exception
# 为了 Fabric 正确地反混淆你的崩溃报告, 你需要从你的
ProGuard 配置中移除这一行:
# -printmapping mapping.txt

# Picasso
-不警告 com.squareup.okhttp.**

```

```

# Support library
-dontwarn android.support.**
-dontwarn android.support.v4.**
-keep class android.support.v4.** { *; }
-keep interface android.support.v4.** { *; }
-dontwarn android.support.v7.**
-keep class android.support.v7.** { *; }
-keep interface android.support.v7.** { *; }

# support design
-dontwarn android.support.design.**
-keep class android.support.design.** { *; }
-keep interface android.support.design.** { *; }
-keep public class android.support.design.R$* { *; }

# retrofit
-dontwarn okio.**
-keepattributes Signature
-keepattributes *Annotation*
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }
-dontwarn com.squareup.okhttp.**

-dontwarn rx.**
-dontwarn retrofit.**
-keep class retrofit.** { *; }
-keepclasseswithmembers class * {
    @retrofit.http.* <methods>;
}

-keep class sun.misc.Unsafe { *; }
#your package path where your gson models are stored
-keep class com.abc.model.** { *; }

# Keep these for GSON and Jackson
-keepattributes Signature
-keepattributes *Annotation*
-keepattributes EnclosingMethod
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.** { *; }

#keep otto
-keepattributes *Annotation*
-keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}

# Crashlytics 2.+
-keep class com.crashlytics.** { *; }
-keep class com.crashlytics.android.**
-keepattributes SourceFile, LineNumberTable, *Annotation*
# If you are using custom exceptions, add this line so that custom exception types are skipped
during obfuscation:
-keep public class * extends java.lang.Exception
# For Fabric to properly de-obfuscate your crash reports, you need to remove this line from your
ProGuard config:
# -printmapping mapping.txt

# Picasso
-dontwarn com.squareup.okhttp.**

```

```

# Volley
-保留 class com.android.volley.toolbox.ImageLoader { *; }

# OkHttp3
-保留 class okhttp3.{**} { *; }
-保留 interface okhttp3.{**} { *; }
-不警告 okhttp3.{**}

# 需要为了 Parcelable/SafeParcelable 创建器不被剥离
-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}

```

第131.2节：在构建时移除跟踪日志（及其他）语句

如果你想移除对某些方法的调用，假设它们返回void且没有副作用（即调用它们不会改变任何系统值、引用参数、静态变量等），那么你可以让ProGuard在构建完成后将它们从输出中移除。

例如，我发现这在移除调试/详细日志语句时很有用，这些语句在调试时有用，但在生产环境中生成它们的字符串是没有必要的。

```

# 移除调试和详细级别的日志语句。
# 这意味着生成这些方法参数的代码也不会被调用。
# 仅当ProGuard配置中未使用-dontoptimize时有效
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}

```

注意：如果在任何ProGuard配置中使用了-dontoptimize，导致不进行代码压缩/移除未使用代码，那么这些语句将不会被剥离。（但谁不想移除未使用的代码呢，对吧？）

注意2：此调用将移除对日志的调用，但不会保护你的代码。字符串实际上仍会保留在生成的apk中。详情请阅读这篇文章。

第131.3节：保护你的代码免受黑客攻击

混淆通常被认为是代码保护的魔法解决方案，通过使你的代码更难理解如果它被黑客反编译的话。

但如果你认为删除Log.x(..)实际上移除了黑客所需的信息，你将会遭遇一个糟糕的惊喜。

删除所有的日志调用：

```

-assumenosideeffects class android.util.Log {
    public static *** d(...);
...等
}

```

确实会移除Log调用本身，但通常不会移除你放入其中的字符串。

例如，如果你在日志调用中输入一个常见的日志消息，如：Log.d(MyTag, "Score="+score);，编译器会将+转换为Log调用外的'new StringBuilder()'。ProGuard不会更改这个新对象。

```

# Volley
-keep class com.android.volley.toolbox.ImageLoader { *; }

# OkHttp3
-keep class okhttp3.{**} { *; }
-keep interface okhttp3.{**} { *; }
-dontwarn okhttp3.{**}

```

```

# Needed for Parcelable/SafeParcelable Creators to not get stripped
-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}

```

Section 131.2: Remove trace logging (and other) statements at build time

If you want to remove calls to certain methods, assuming they return void and have no side affects (as in, calling them doesn't change any system values, reference arguments, statics, etc.) then you can have ProGuard remove them from the output after the build is complete.

For example, I find this useful in removing debug/verbose logging statements useful in debugging, but generating the strings for them is unnecessary in production.

```

# Remove the debug and verbose level Logging statements.
# That means the code to generate the arguments to these methods will also not be called.
# ONLY WORKS IF -dontoptimize IS _NOT_ USED in any ProGuard configs
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}

```

Note: If -dontoptimize is used in any ProGuard config so that it is not minifying/removing unused code, then this will not strip out the statements. (But who would not want to remove unused code, right?)

Note2: this call will remove the call to log, but will not protect your code. The Strings will actually remain in the generated apk. Read more in this post.

Section 131.3: Protecting your code from hackers

Obfuscation is often considered as a magic solution for code protection, by making your code harder to understand if it ever gets de-compiled by hackers.

But if you're thinking that removing the Log.x(..) actually removes the information the hackers need, you'll have a nasty surprise.

Removing all your log calls with:

```

-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}

```

will indeed remove the Log call itself, but usually *not* the Strings you put into them.

If for example inside your log call you type a common log message such as: Log.d(MyTag, "Score="+score);, the compiler converts the + to a 'new StringBuilder()' outside the Log call. ProGuard doesn't change this new object.

你的反编译代码仍然会有一个悬挂的StringBuilder用于"Score="，后面附加了被混淆的

score变量的版本（假设它被转换成了b）。

现在黑客知道了什么是b，并能理解你的代码。

一个实际移除代码中这些残留的好方法是，要么一开始就不要放进去

（改用字符串格式化器，并配合proguard规则移除它们），要么用以下方式包裹你的Log调用：

```
if (BuildConfig.DEBUG) {  
    Log.d(TAG, "...+var);  
}
```

提示：

通过自己反编译测试你的混淆代码保护得有多好！

1. [dex2jar](#) - 将apk转换为jar

jd2... - 反编译jar并在图形界面编辑器中打开

第131.4节：为你的构建启用ProGuard

要为你的应用启用ProGuard配置，需要在模块级gradle文件中启用它。你

需要将minifyEnabled设置为true。

你也可以启用shrinkResources为true，这将移除ProGuard标记为未使用的资源。

```
buildTypes {  
    release {  
        minifyEnabled true  
        shrinkResources true  
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
    }  
}
```

上述代码将应用包含在proguard-rules.pro（Eclipse中的 "proguard-project.txt"）中的ProGuard配置到你的发布apk中。

为了使你能够在堆栈跟踪中确定异常发生的行，"proguard-rules.pro"

应包含以下内容：

```
-renamesourcefileattribute SourceFile  
-keepattributes SourceFile,LineNumberTable
```

要在Eclipse中启用Proguard，请将proguard.config=\${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt添加到 "project.properties"

第131.5节：使用自定义混淆配置文件启用ProGuard

ProGuard允许开发者对代码进行混淆、缩减和优化。

#1 该流程的第一步是启用构建中的proguard。

这可以通过将所需构建的 'minifyEnabled' 命令设置为 true 来完成

#2 第二步是指定我们为给定构建使用哪些proguard文件

Your de-compiled code will still have a hanging StringBuilder for "Score=", appended with the obfuscated version for score variable (let's say it was converted to b).

Now the hacker knows what is b, and make sense of your code.

A good practice to actually remove these residuals from your code is either not put them there in the first place (Use String formatter instead, with proguard rules to remove them), or to wrap your Log calls with:

```
if (BuildConfig.DEBUG) {  
    Log.d(TAG, "...+var);  
}
```

Tip:

Test how well protected your obfuscated code is by de-compiling it yourself!

1. [dex2jar](#) - converts the apk to jar
2. [jd](#) - decompiles the jar and opens it in a gui editor

Section 131.4: Enable ProGuard for your build

For enabling ProGuard configurations for your application you need to enable it in your module level gradle file. you need to set the value of minifyEnabled true.

You can also enable shrinkResources true which will remove resources that ProGuard flags as unused.

```
buildTypes {  
    release {  
        minifyEnabled true  
        shrinkResources true  
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
    }  
}
```

The above code will apply your ProGuard configurations contained in proguard-rules.pro ("proguard-project.txt" in Eclipse) to your released apk.

To enable you to later determine the line on which an exception occurred in a stack trace, "proguard-rules.pro" should contain following lines:

```
-renamesourcefileattribute SourceFile  
-keepattributes SourceFile,LineNumberTable
```

To enable Proguard in Eclipse add proguard.config=\${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt to "project.properties"

Section 131.5: Enabling ProGuard with a custom obfuscation configuration file

ProGuard allows the developer to obfuscate, shrink and optimize his code.

#1 The first step of the procedure is to enable proguard on the build.

This can be done by setting the 'minifyEnabled' command to true on your desired build

#2 The second step is to specify which proguard files are we using for the given build

这可以通过将 'proguardFiles' 行设置为正确的文件名来完成

```
buildTypes {  
    debug {  
        minifyEnabled false  
    }  
    testRelease {  
        minifyEnabled true  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-  
        tests.pro'  
    }  
    productionRelease {  
        minifyEnabled true  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-  
        tests.pro', 'proguard-rules-release.pro'  
    }  
}
```

#3 开发者随后可以使用他所需的规则编辑他的proguard文件。

这可以通过编辑文件（例如 'proguard-rules-tests.pro'）并添加所需的约束来完成。

下面的文件作为一个示例proguard文件

```
// 默认和基本的优化配置  
-optimizationpasses 5  
-dontpreverify  
-repackageclasses "  
-allowaccessmodification  
-optimizations !code/simplification/arithmetic  
-保留属性 *Annotation*  
  
-verbose  
  
-dump混淆/class_files.txt  
-printseeds混淆/seeds.txt  
-printusage混淆/unused.txt // 在过程中被剔除的未使用类  
-printmapping混淆/mapping.txt // 显示应用 ProGuard 后类的混淆名称的映射文件  
  
// 开发者可以为混淆指定关键词（我自己偶尔会用水果作为混淆名称:-) )  
-obfuscationdictionary混淆/keywords.txt  
-classobfuscationdictionary混淆/keywords.txt  
-packageobfuscationdictionary混淆/keywords.txt
```

最后，每当开发者运行和/或生成新的 .APK 文件时，自定义的 ProGuard 配置将被应用，从而满足需求。

This can be done by setting the 'proguardFiles' line with the proper filenames

```
buildTypes {  
    debug {  
        minifyEnabled false  
    }  
    testRelease {  
        minifyEnabled true  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-  
        tests.pro'  
    }  
    productionRelease {  
        minifyEnabled true  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-  
        tests.pro', 'proguard-rules-release.pro'  
    }  
}
```

#3 The developer can then edit his proguard file with the rules he desires.

That can be done by editting the file (for example 'proguard-rules-tests.pro') and adding the desired constraints.
The following file serves as an example proguard file

```
// default & basic optimization configurations  
-optimizationpasses 5  
-dontpreverify  
-repackageclasses ''  
-allowaccessmodification  
-optimizations !code/simplification/arithmetic  
-keepattributes *Annotation*  
  
-verbose  
  
-dump obfuscation/class_files.txt  
-printseeds obfuscation/seeds.txt  
-printusage obfuscation/unused.txt // unused classes that are stripped out in the process  
-printmapping obfuscation/mapping.txt // mapping file that shows the obfuscated names of the classes  
after proguard is applied  
  
// the developer can specify keywords for the obfuscation (I myself use fruits for obfuscation names  
once in a while :-))  
-obfuscationdictionary obfuscation/keywords.txt  
-classobfuscationdictionary obfuscation/keywords.txt  
-packageobfuscationdictionary obfuscation/keywords.txt
```

Finally, whenever the developer runs and/or generates his new .APK file, the custom proguard configurations will be applied thus fulfilling the requirements.

第132章：类型定义注释： @IntDef, @StringDef

第132.1节：IntDef 注解

该注解确保只使用您期望的有效整数常量。

下面的示例说明了创建注解的步骤：

```
import android.support.annotation.IntDef;

public abstract class 车辆 {

    //定义接受的常量列表
    @IntDef({微型车, 敞篷车, 超级跑车, 小型面包车, 运动型多功能车})

    //告诉编译器不要在.class文件中存储注解数据
    @Retention(RetentionPolicy.SOURCE)
    //声明 CarType 注解
    public @interface CarType {}

    //声明常量
    public static final int MICROCAR = 0;
    public static final int CONVERTIBLE = 1;
    public static final int SUPERCAR = 2;
    public static final int MINIVAN = 3;
    public static final int SUV = 4;

    @CarType
    private int mType;

    @CarType
    public int getCarType(){
        return mType;
    }

    public void setCarType(@CarType int type){
        mType = type;
    }
}
```

它们还支持代码补全，自动提供允许的常量。

当你编译这段代码时，如果类型参数未引用已定义的常量之一，将会生成警告。

第132.2节：使用标志组合常量

使用 IntDef#flag() 属性设置为 true，可以组合多个常量。

使用本主题中的相同示例：

```
public abstract class 车 {

    //定义接受的常量列表
    @IntDef(flag=true, value={微型车, 敞篷车, 超级跑车, 多功能车, 运动型多用途车})

    //告诉编译器不要在.class文件中存储注解数据
    @Retention(RetentionPolicy.SOURCE)
```

Chapter 132: Typedef Annotations: @IntDef, @StringDef

Section 132.1: IntDef Annotations

This [annotation](#) ensures that only the valid integer constants that you expect are used.

The following example illustrates the steps to create an annotation:

```
import android.support.annotation.IntDef;

public abstract class Car {

    //Define the list of accepted constants
    @IntDef({MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)
    //Declare the CarType annotation
    public @interface CarType {}

    //Declare the constants
    public static final int MICROCAR = 0;
    public static final int CONVERTIBLE = 1;
    public static final int SUPERCAR = 2;
    public static final int MINIVAN = 3;
    public static final int SUV = 4;

    @CarType
    private int mType;

    @CarType
    public int getCarType(){
        return mType;
    }

    public void setCarType(@CarType int type){
        mType = type;
    }
}
```

They also enable code completion to automatically offer the allowed constants.

When you build this code, a warning is generated if the type parameter does not reference one of the defined constants.

Section 132.2: Combining constants with flags

Using the IntDef#flag() attribute set to `true`, multiple constants can be combined.

Using the same example in this topic:

```
public abstract class Car {

    //Define the list of accepted constants
    @IntDef(flag=true, value={MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)
```

....

}

用户可以使用标志（例如 |, &, ^）组合允许的常量。



....

}

Users can combine the allowed constants with a flag (such as |, &, ^).

第133章：截取屏幕截图

第133.1节：截取特定视图的屏幕截图

如果你想截取特定视图 v 的屏幕截图，可以使用以下代码：

```
Bitmap viewBitmap = Bitmap.createBitmap(v.getWidth(), v.getHeight(), Bitmap.Config.RGB_565);
Canvas viewCanvas = new Canvas(viewBitmap);
Drawable backgroundDrawable = v.getBackground();

if(backgroundDrawable != null){
    // 将背景绘制到画布上。
    backgroundDrawable.draw(viewCanvas);
}
else{
    viewCanvas.drawColor(Color.GREEN);
    // 将视图绘制到画布上。
    v.draw(viewCanvas)
}

// 将上面生成的位图写入文件。
String fileStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
OutputStream outputStream = null;
try{
    imgFile = new
    File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), fileStamp +
    ".png");
    outputStream = new FileOutputStream(imgFile);
    viewBitmap.compress(Bitmap.CompressFormat.PNG, 40, outputStream);
    outputStream.close();
}
catch(Exception e){
    e.printStackTrace();
}
```

第133.2节：通过Android Studio截取屏幕截图

1. 打开Android监视器标签
2. 点击屏幕截图按钮

Chapter 133: Capturing Screenshots

Section 133.1: Taking a screenshot of a particular view

If you want to take a screenshot of a particular View v, then you can use the following code:

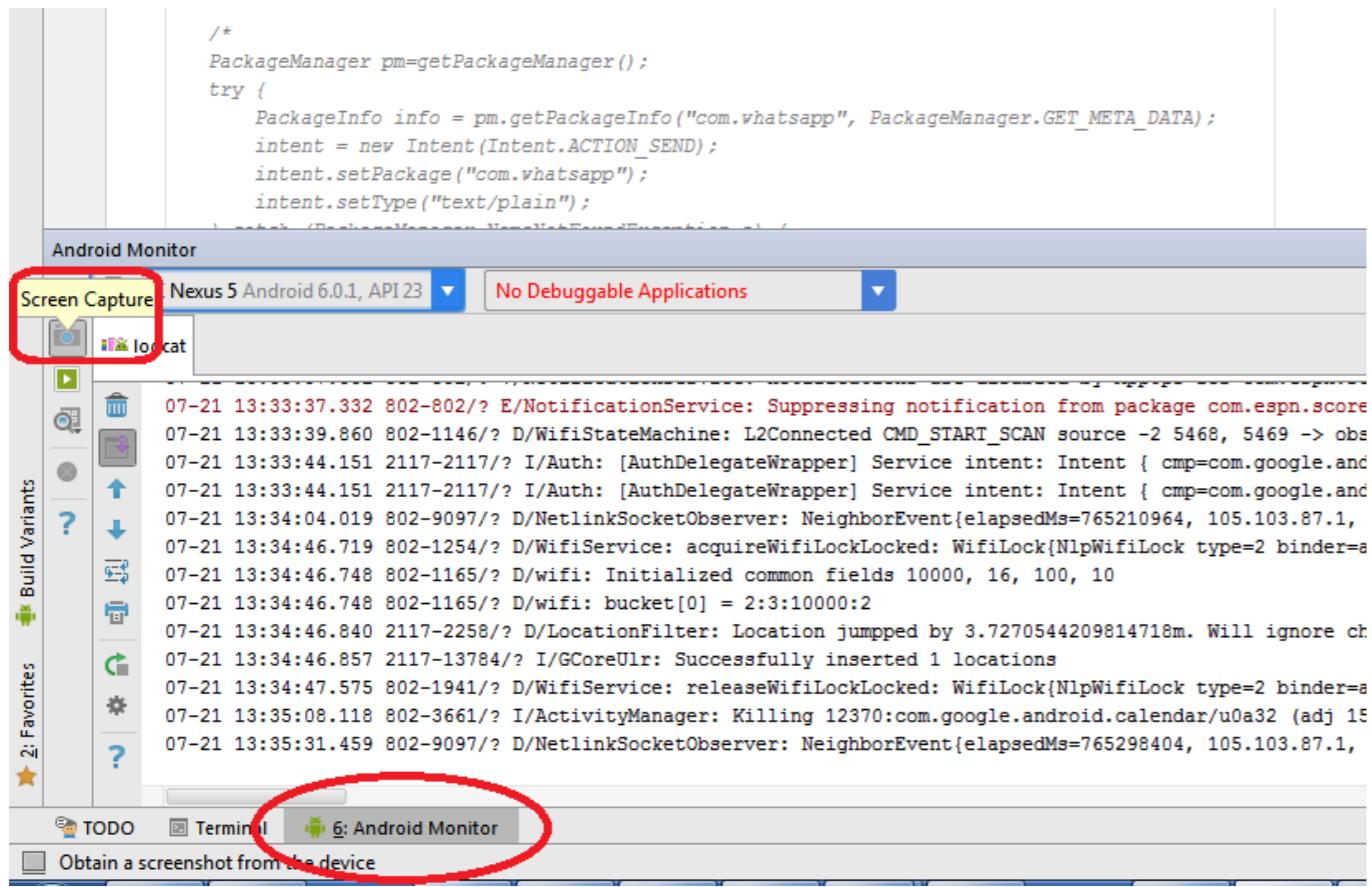
```
Bitmap viewBitmap = Bitmap.createBitmap(v.getWidth(), v.getHeight(), Bitmap.Config.RGB_565);
Canvas viewCanvas = new Canvas(viewBitmap);
Drawable backgroundDrawable = v.getBackground();

if(backgroundDrawable != null){
    // Draw the background onto the canvas.
    backgroundDrawable.draw(viewCanvas);
}
else{
    viewCanvas.drawColor(Color.GREEN);
    // Draw the view onto the canvas.
    v.draw(viewCanvas)
}

// Write the bitmap generated above into a file.
String fileStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
OutputStream outputStream = null;
try{
    imgFile = new
    File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), fileStamp +
    ".png");
    outputStream = new FileOutputStream(imgFile);
    viewBitmap.compress(Bitmap.CompressFormat.PNG, 40, outputStream);
    outputStream.close();
}
catch(Exception e){
    e.printStackTrace();
}
```

Section 133.2: Capturing Screenshot via Android Studio

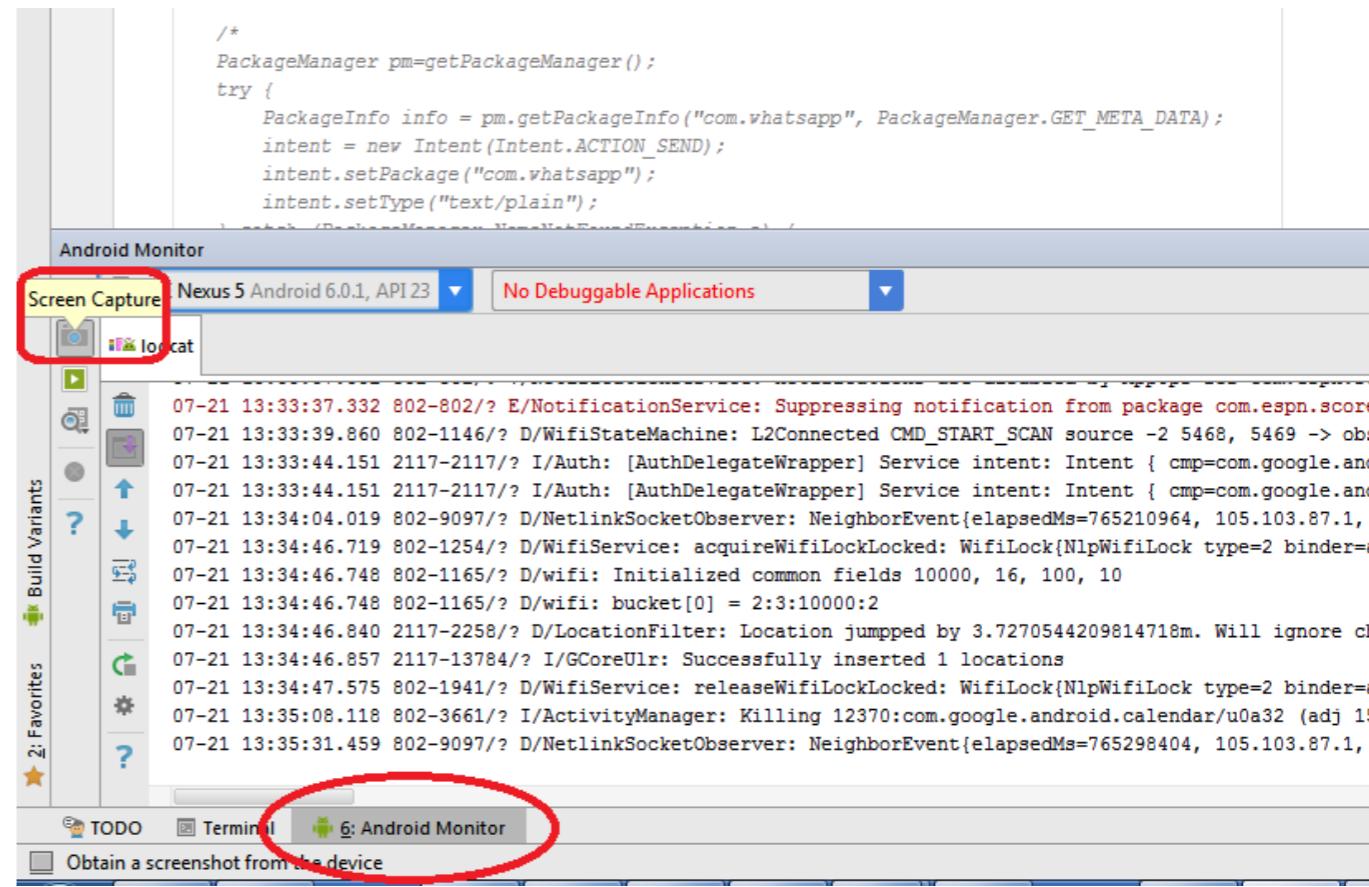
1. Open Android Monitor Tab
2. Click on Screen Capture Button



第133.3节：通过ADB截取屏幕截图并直接保存到电脑

如果你使用Linux（或带有Cygwin的Windows），你可以运行：

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```



Section 133.3: Capturing Screenshot via ADB and saving directly in your PC

If you use Linux (or Windows with Cygwin), you can run:

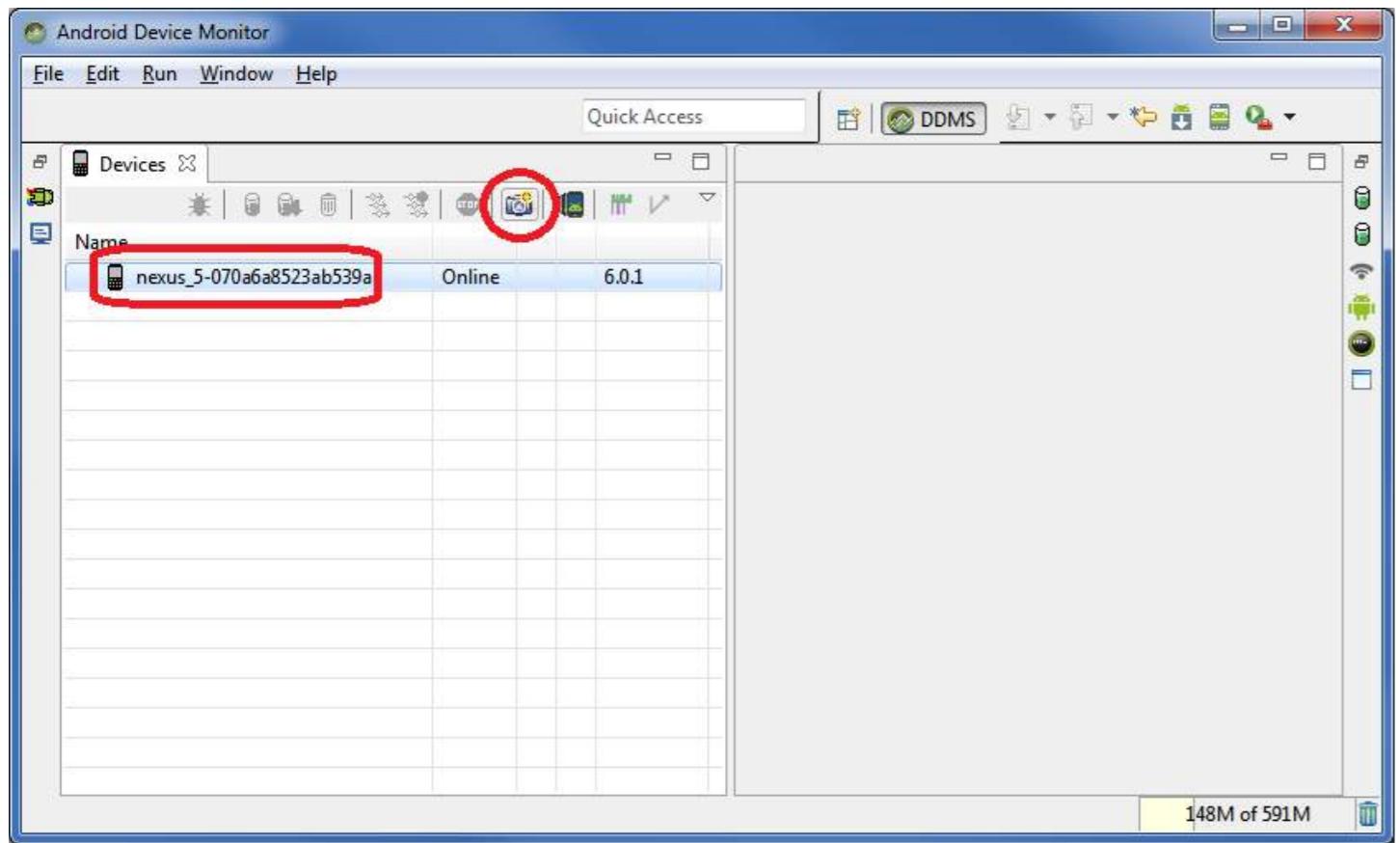
```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

第133.4节：通过Android设备监视器截取屏幕截图

1. 打开Android设备监视器（即 `C:<ANDROID_SDK_LOCATION>\tools\monitor.bat`）
2. 选择您的设备
3. 点击屏幕截图按钮

Section 133.4: Capturing Screenshot via Android Device Monitor

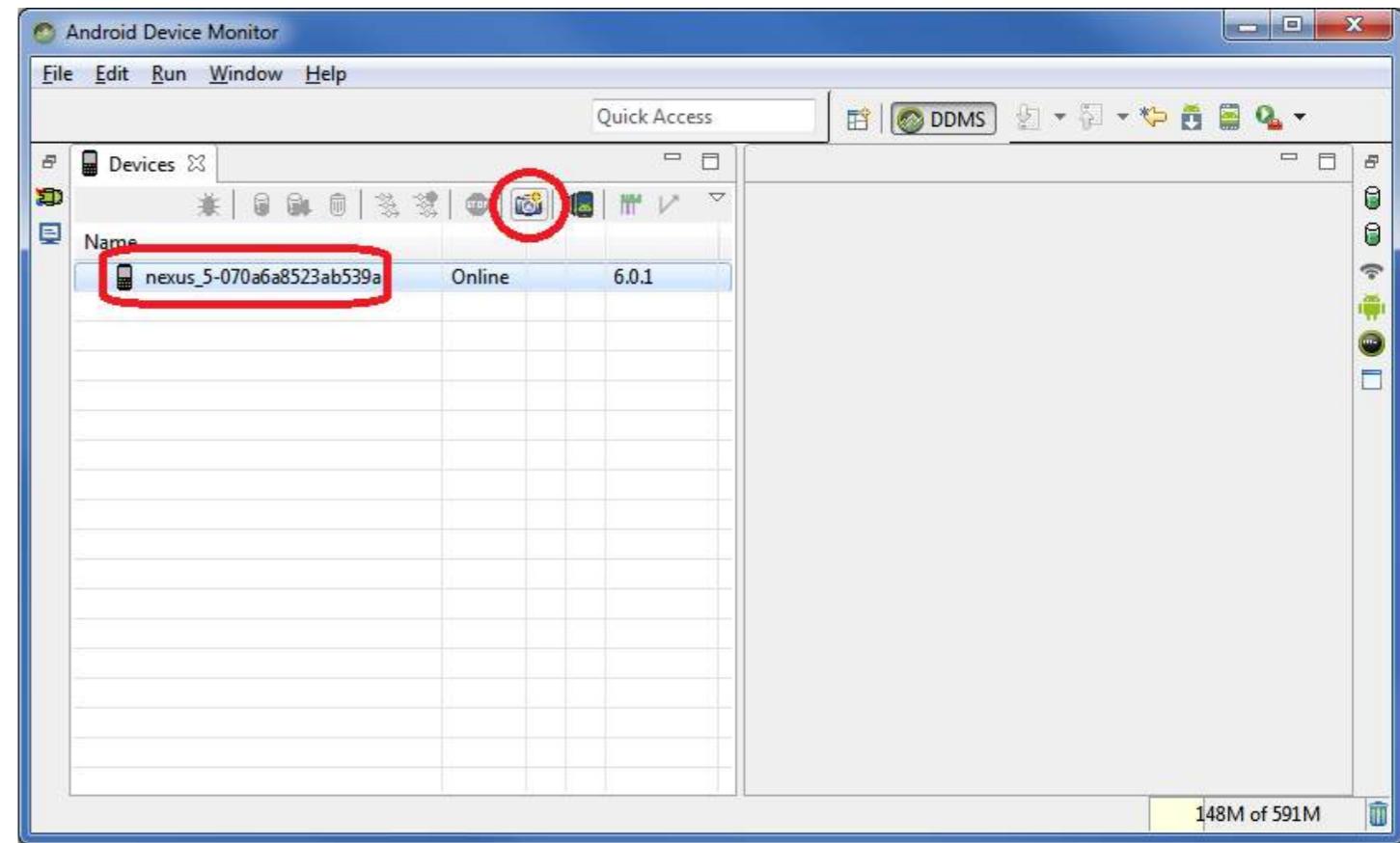
1. Open Android Device Monitor (*i.e.* `C:<ANDROID_SDK_LOCATION>\tools\monitor.bat`)
2. Select your device
3. Click on Screen Capture Button



第133.5节：通过ADB捕获屏幕截图

下面的示例将屏幕截图保存到设备的内部存储中。

```
adb shell screencap /sdcard/screen.png
```



Section 133.5: Capturing Screenshot via ADB

Example below saves a screenshot on Devices's Internal Storage.

```
adb shell screencap /sdcard/screen.png
```

第134章：MVP架构

本主题将通过各种示例介绍Android的模型-视图-主持人（MVP）架构。

第134.1节：模型视图主持人（MVP）模式中的登录示例

让我们通过一个简单的登录界面来看看MVP的实际应用。有两个按钮——一个用于登录操作，另一个用于注册界面；两个编辑框——一个用于邮箱，另一个用于密码。

登录片段（视图）

```
public class LoginFragment extends Fragment implements LoginContract.PresenterToView,
View.OnClickListener {

    private View view;
    private EditText email, password;
    private Button login, register;

    private LoginContract.ToPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        email = (EditText) view.findViewById(R.id.email_et);
        password = (EditText) view.findViewById(R.id.password_et);
        login = (Button) view.findViewById(R.id.login_btn);
        login.setOnClickListener(this);
        register = (Button) view.findViewById(R.id.register_btn);
        register.setOnClickListener(this);

        presenter = new LoginPresenter(this);

        presenter.isLoggedIn();
    }

    @Override
    public void onLoginResponse(boolean isLoggedIn) {
        if (isLoggedIn) {
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onError(String message) {
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void isLoggedIn(boolean isLoggedIn) {
        if (isLoggedIn) {
```

Chapter 134: MVP Architecture

This topic will provide [Model-View-Presenter \(MVP\)](#) architecture of Android with various examples.

Section 134.1: Login example in the Model View Presenter (MVP) pattern

Let's see MVP in action using a simple Login Screen. There are two [Buttons](#)—one for login action and another for a registration screen; two [EditTexts](#)—one for the email and the other for the password.

LoginFragment (The View)

```
public class LoginFragment extends Fragment implements LoginContract.PresenterToView,
View.OnClickListener {

    private View view;
    private EditText email, password;
    private Button login, register;

    private LoginContract.ToPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        email = (EditText) view.findViewById(R.id.email_et);
        password = (EditText) view.findViewById(R.id.password_et);
        login = (Button) view.findViewById(R.id.login_btn);
        login.setOnClickListener(this);
        register = (Button) view.findViewById(R.id.register_btn);
        register.setOnClickListener(this);

        presenter = new LoginPresenter(this);

        presenter.isLoggedIn();
    }

    @Override
    public void onLoginResponse(boolean isLoggedIn) {
        if (isLoggedIn) {
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onError(String message) {
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void isLoggedIn(boolean isLoggedIn) {
        if (isLoggedIn) {
```

```

startActivity(new Intent(getActivity(), MapActivity.class));
    getActivity().finish();
}
}

@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.login_btn:
LoginItem loginItem = new LoginItem();
            loginItem.setPassword(password.getText().toString().trim());
            loginItem.setEmail(email.getText().toString().trim());
presenter.login(loginItem);
            break;
        case R.id.register_btn:
startActivity(new Intent(getActivity(), RegisterActivity.class));
            getActivity().finish();
            break;
    }
}
}

```

登录演示者 (The Presenter)

```

public class LoginPresenter implements LoginContract.ToPresenter {

private LoginContract.PresenterToModel model;
private LoginContract.PresenterToView view;

public LoginPresenter(LoginContract.PresenterToView view) {
    this.view = view;
model = new LoginModel(this);
}

@Override
public void login(LoginItem 用户凭证) {
    model.login(用户凭证);
}

@Override
public void isLoggedIn() {
    model.isLoggedIn();
}

@Override
public void onLoginResponse(boolean 登录成功) {
    view.onLoginResponse(登录成功);
}

@Override
public void onError(String 消息) {
    view.onError(消息);
}

@Override
public void isloggedIn(boolean 已登录) {
    view.isLoggedIn(已登录);
}
}

```

LoginModel (模型)

```

startActivity(new Intent(getActivity(), MapActivity.class));
    getActivity().finish();
}
}

@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.login_btn:
LoginItem loginItem = new LoginItem();
            loginItem.setPassword(password.getText().toString().trim());
            loginItem.setEmail(email.getText().toString().trim());
presenter.login(loginItem);
            break;
        case R.id.register_btn:
startActivity(new Intent(getActivity(), RegisterActivity.class));
            getActivity().finish();
            break;
    }
}
}

```

LoginPresenter (The Presenter)

```

public class LoginPresenter implements LoginContract.ToPresenter {

private LoginContract.PresenterToModel model;
private LoginContract.PresenterToView view;

public LoginPresenter(LoginContract.PresenterToView view) {
    this.view = view;
    model = new LoginModel(this);
}

@Override
public void login(LoginItem userCredentials) {
    model.login(userCredentials);
}

@Override
public void isLoggedIn() {
    model.isLoggedIn();
}

@Override
public void onLoginResponse(boolean isLoginSuccess) {
    view.onLoginResponse(isLoginSuccess);
}

@Override
public void onError(String message) {
    view.onError(message);
}

@Override
public void isloggedIn(boolean isLoggedin) {
    view.isLoggedIn(isLoggedin);
}
}

```

LoginModel (The Model)

```

public class LoginModel implements LoginContract.PresenterToModel,
ResponseErrorListener.ErrorListener {

    private static final String TAG = LoginModel.class.getSimpleName();
    private LoginContract.ToPresenter presenter;

    public LoginModel(LoginContract.ToPresenter presenter) {
        this.presenter = presenter;
    }

    @Override
    public void login(LoginItem userCredentials) {
        if (validateData(userCredentials)) {
            try {
                performLoginOperation(userCredentials);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            presenter.onError(BaseContext.getContext().getString(R.string.error_login_field_validation));
        }
    }

    @Override
    public void isLoggedIn() {
        DatabaseHelper database = new DatabaseHelper(BaseContext.getContext());
        presenter.isloggedIn(database.isLoggedIn());
    }

    private boolean validateData(LoginItem userCredentials) {
        return Patterns.EMAIL_ADDRESS.matcher(userCredentials.getEmail()).matches()
            && !userCredentials.getPassword().trim().equals("");
    }

    private void performLoginOperation(final LoginItem userCredentials) throws JSONException {

        JSONObject postData = new JSONObject();
        postData.put(Constants.EMAIL, userCredentials.getEmail());
        postData.put(Constants.PASSWORD, userCredentials.getPassword());

        JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, Url.AUTH, postData,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        String token = response.getString(Constants.ACCESS_TOKEN);
                        DatabaseHelper databaseHelper = new
                            DatabaseHelper(BaseContext.getContext());
                        databaseHelper.login(token);
                        Log.d(TAG, "onResponse: " + token);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    presenter.onLoginResponse(true);
                }
            }, newErrorResponse(this));

        RequestQueue queue = Volley.newRequestQueue(BaseContext.getContext());
        queue.add(request);
    }
}

```

```

public class LoginModel implements LoginContract.PresenterToModel,
ResponseErrorListener.ErrorListener {

    private static final String TAG = LoginModel.class.getSimpleName();
    private LoginContract.ToPresenter presenter;

    public LoginModel(LoginContract.ToPresenter presenter) {
        this.presenter = presenter;
    }

    @Override
    public void login(LoginItem userCredentials) {
        if (validateData(userCredentials)) {
            try {
                performLoginOperation(userCredentials);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            presenter.onError(BaseContext.getContext().getString(R.string.error_login_field_validation));
        }
    }

    @Override
    public void isLoggedIn() {
        DatabaseHelper database = new DatabaseHelper(BaseContext.getContext());
        presenter.isloggedIn(database.isLoggedIn());
    }

    private boolean validateData(LoginItem userCredentials) {
        return Patterns.EMAIL_ADDRESS.matcher(userCredentials.getEmail()).matches()
            && !userCredentials.getPassword().trim().equals("");
    }

    private void performLoginOperation(final LoginItem userCredentials) throws JSONException {

        JSONObject postData = new JSONObject();
        postData.put(Constants.EMAIL, userCredentials.getEmail());
        postData.put(Constants.PASSWORD, userCredentials.getPassword());

        JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, Url.AUTH, postData,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        String token = response.getString(Constants.ACCESS_TOKEN);
                        DatabaseHelper databaseHelper = new
                            DatabaseHelper(BaseContext.getContext());
                        databaseHelper.login(token);
                        Log.d(TAG, "onResponse: " + token);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    presenter.onLoginResponse(true);
                }
            }, newErrorResponse(this));

        RequestQueue queue = Volley.newRequestQueue(BaseContext.getContext());
        queue.add(request);
    }
}

```

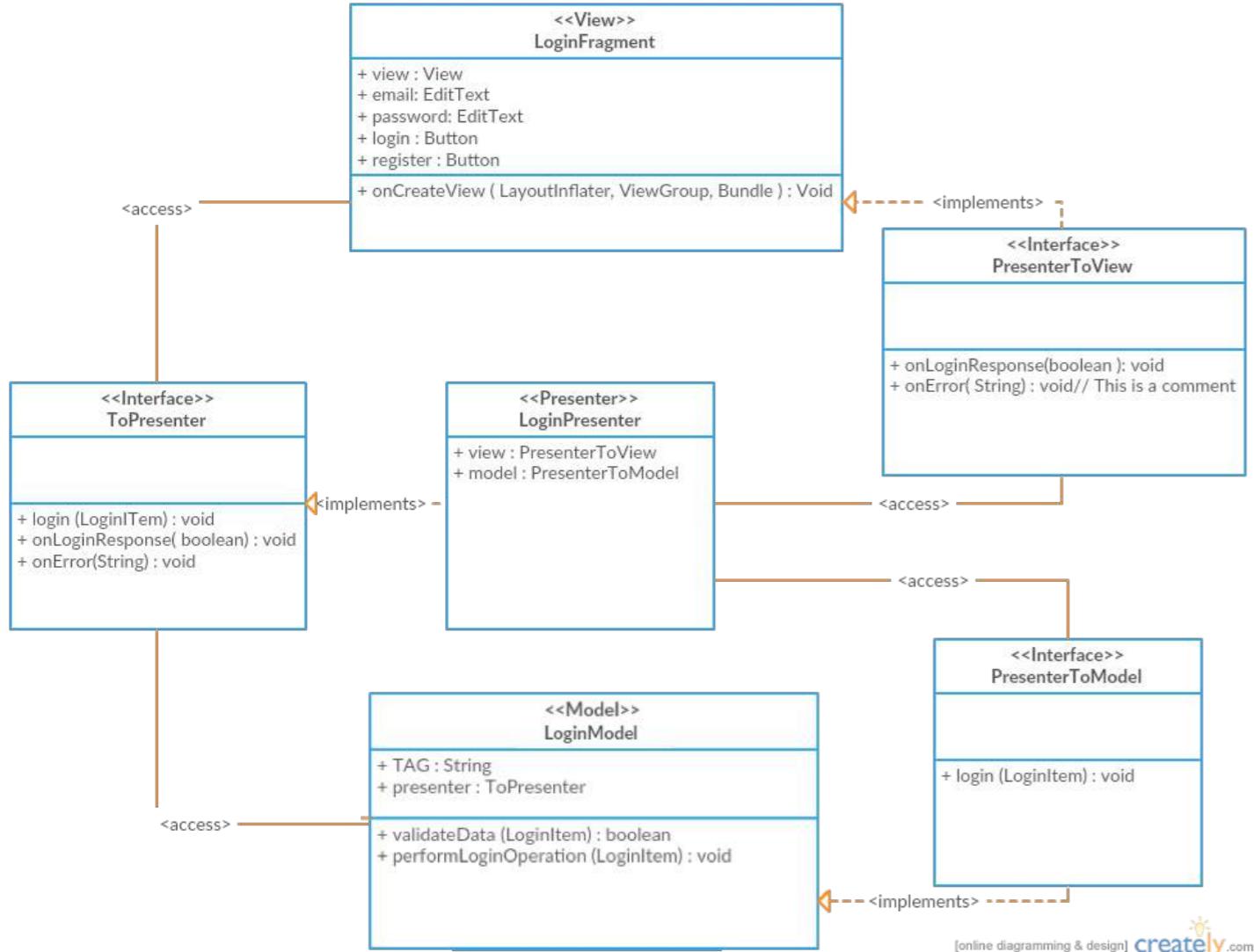
```

@Override
public void onError(String message) {
    presenter.onError(message);
}

```

类图

让我们通过类图来看看这个动作。



注意事项：

- 此示例使用Volley进行网络通信，但MVP不需要此库
- UrlUtils是一个包含我所有API端点链接的类Response.ErrorListener.ErrorLister
- ner是一个接口，用于监听ErrorResponse中的错误，它实现了Volley's Response.ErrorListener；这些类未包含在此处，因为它们不是本示例的直接部分

第134.2节：MVP中的简单登录示例

所需的包结构

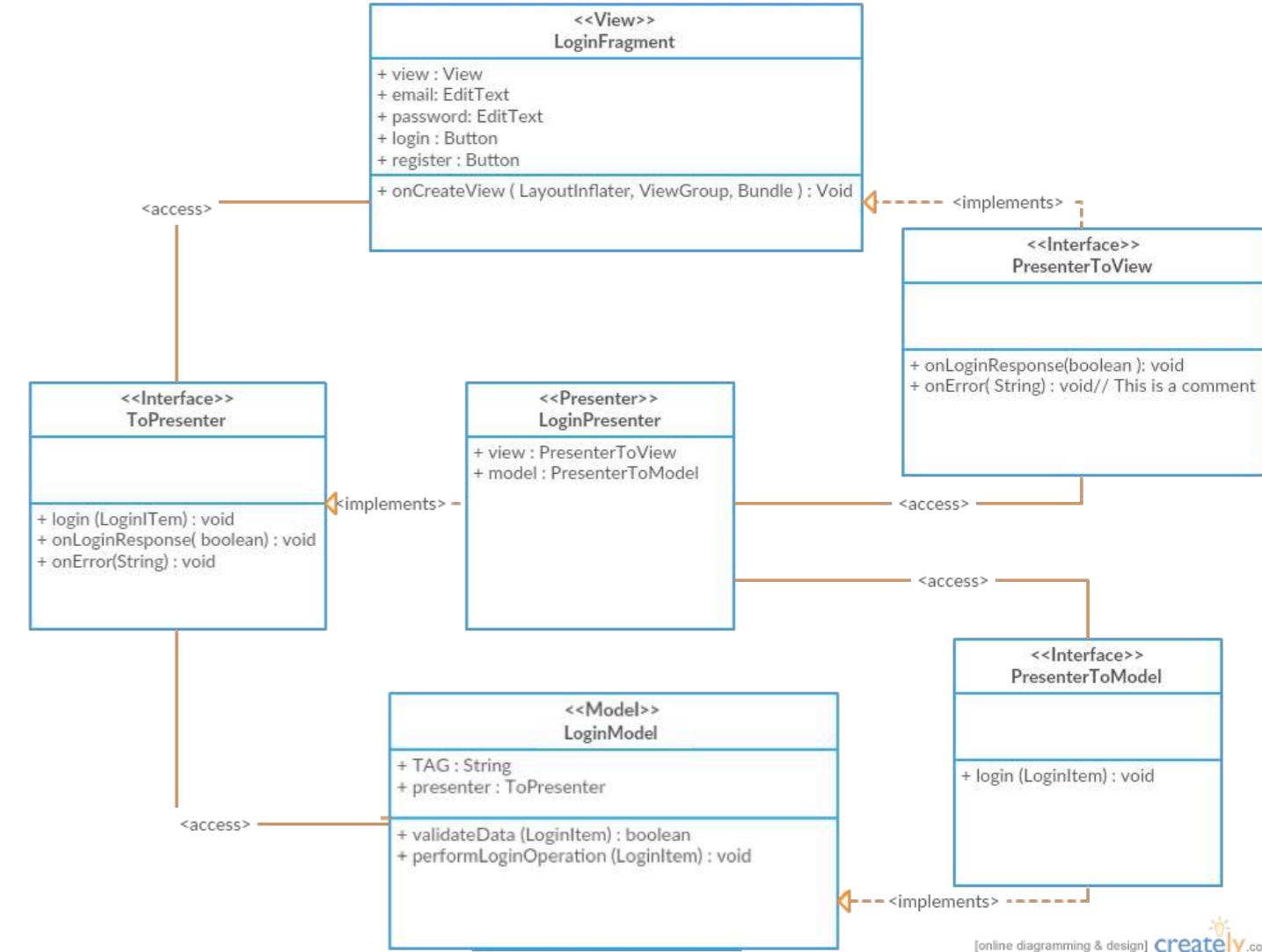
```

@Override
public void onError(String message) {
    presenter.onError(message);
}

```

Class Diagram

Let's see the action in the form of class diagram.

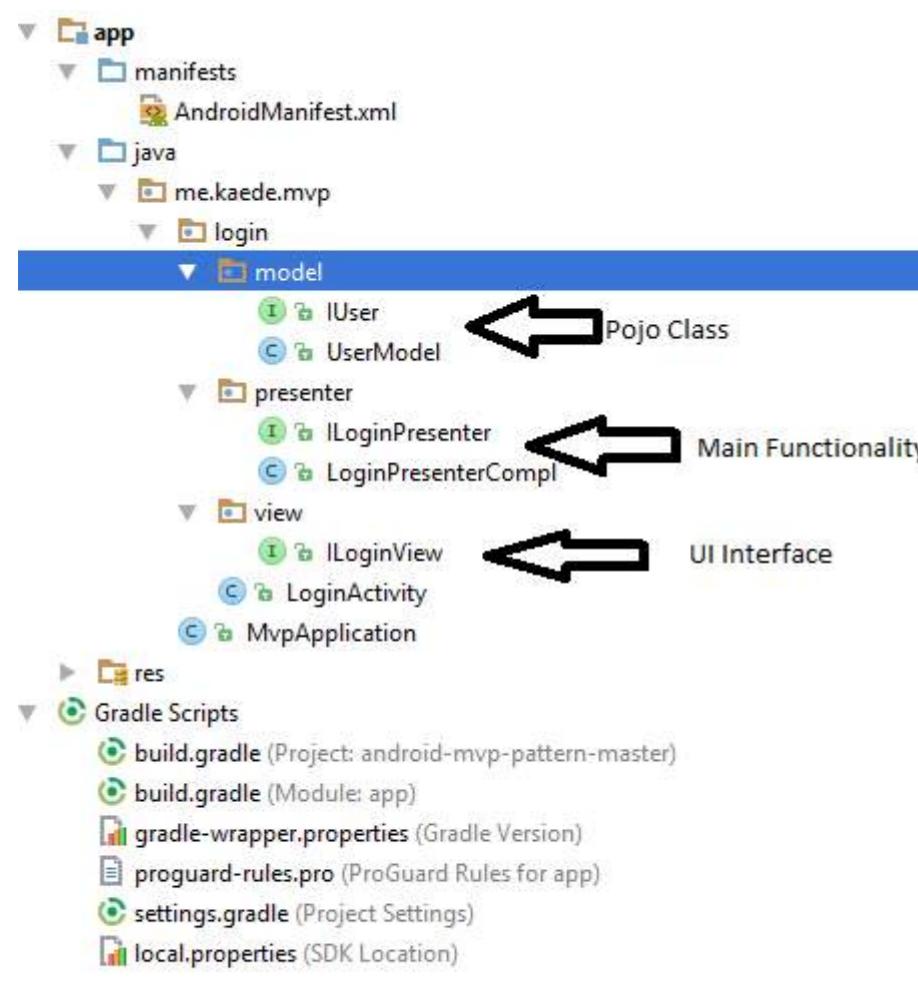


Notes:

- This example uses [Volley](#) for network communication, but this library is not required for MVP
- UrlUtils is a class which contains all the links for my API Endpoints
- [Response.ErrorListener.ErrorLister](#) is an **interface** that listens for error in ErrorResponse that **implements** [Volley's Response.ErrorListener](#); these classes are not included here as they are not directly part of this example

Section 134.2: Simple Login Example in MVP

Required package structure



XML activity_login

```

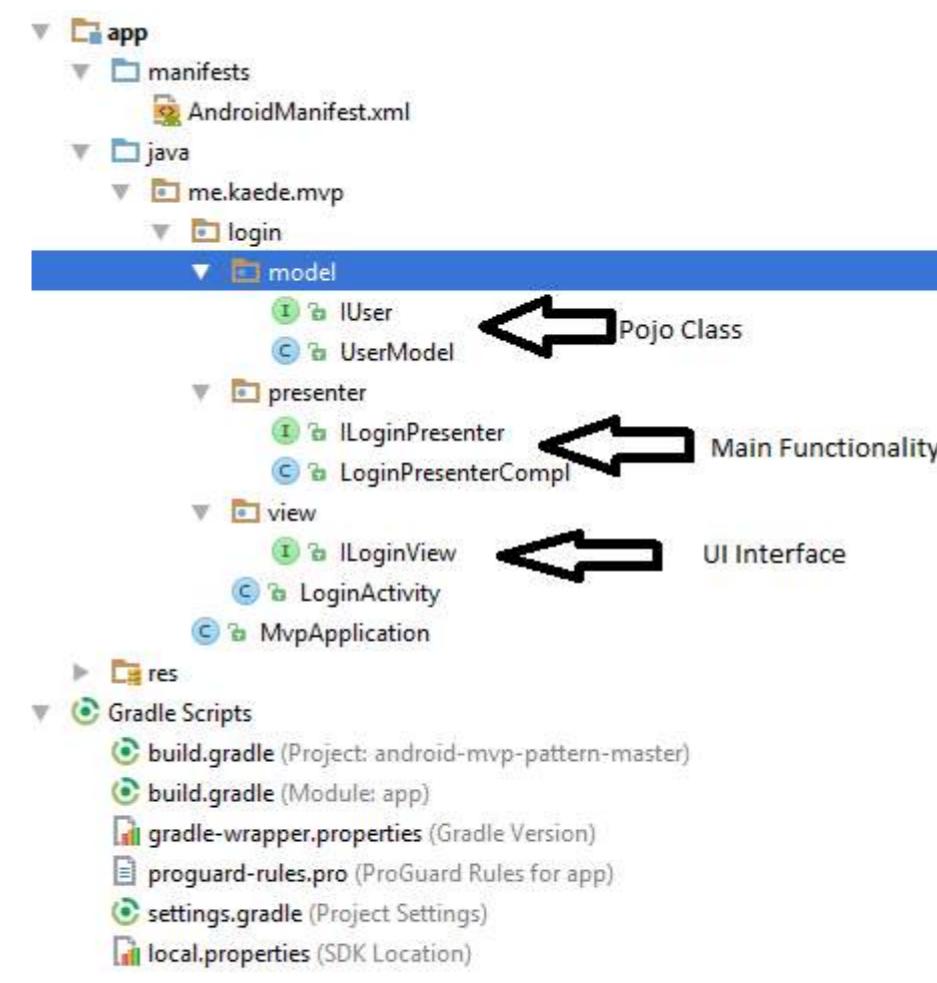
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
        android:id="@+id/et_login_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="用户名" />

    <EditText
        android:id="@+id/et_login_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="密码" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

```



XML activity_login

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
        android:id="@+id/et_login_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="USERNAME" />

    <EditText
        android:id="@+id/et_login_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="PASSWORD" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

```

```

<Button
    android:id="@+id/btn_login_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginRight="4dp"
    android:layout_weight="1"
    android:text="登录" />

<Button
    android:id="@+id/btn_login_clear"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="4dp"
    android:layout_weight="1"
    android:text="清除" />
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:text="正确用户 : mvp, mvp" />

<ProgressBar
    android:id="@+id/progress_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp" />

</LinearLayout>

```

活动类 LoginActivity.class

```

public class 登录活动 extends AppCompatActivity implements ILoginView, View.OnClickListener {
    private 编辑文本 用户编辑;
    private 编辑文本 密码编辑;
    private 按钮 登录按钮;
    private 按钮 清除按钮;
    private ILoginPresenter 登录主持人;
    private 进度条 进度条;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.布局.activity_login);

        //查找视图
        用户编辑 = (编辑文本) this.findViewById(R.id.et_login_username);
        密码编辑 = (编辑文本) this.findViewById(R.id.et_login_password);
        登录按钮 = (按钮) this.findViewById(R.id.btn_login_login);
        清除按钮 = (按钮) this.findViewById(R.id.btn_login_clear);
        进度条 = (进度条) this.findViewById(R.id.progress_login);

        //设置监听器
        登录按钮.setOnClickListener(this);
        清除按钮.setOnClickListener(this);

        //初始化
        登录主持人 = new 登录主持人实现(this);
        登录主持人.setProgressbarVisibility(View.INVISIBLE);
    }

    @Override

```

```

<Button
    android:id="@+id/btn_login_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginRight="4dp"
    android:layout_weight="1"
    android:text="Login" />

<Button
    android:id="@+id/btn_login_clear"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="4dp"
    android:layout_weight="1"
    android:text="Clear" />
</LinearLayout>

```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:text="correct user: mvp, mvp" />

```

```

<ProgressBar
    android:id="@+id/progress_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp" />

</LinearLayout>

```

Activity Class LoginActivity.class

```

public class LoginActivity extends AppCompatActivity implements ILoginView, View.OnClickListener {
    private EditText editUser;
    private EditText editPass;
    private Button btnLogin;
    private Button btnClear;
    private ILoginPresenter loginPresenter;
    private ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        //find view
        editUser = (EditText) this.findViewById(R.id.et_login_username);
        editPass = (EditText) this.findViewById(R.id.et_login_password);
        btnLogin = (Button) this.findViewById(R.id.btn_login_login);
        btnClear = (Button) this.findViewById(R.id.btn_login_clear);
        progressBar = (ProgressBar) this.findViewById(R.id.progress_login);

        //set listener
        btnLogin.setOnClickListener(this);
        btnClear.setOnClickListener(this);

        //init
        loginPresenter = new LoginPresenterCompl(this);
        loginPresenter.setProgressbarVisibility(View.INVISIBLE);
    }

    @Override

```

```

public void onClick(View v) {
    switch (v.getId()) {
        case R.id.btn_login_clear:
            loginPresenter.clear();
            break;
        case R.id.btn_login_login:
            loginPresenter.setProgressBarVisibility(View.VISIBLE);
            btnLogin.setEnabled(false);
            btnClear.setEnabled(false);
            loginPresenter.doLogin(editUser.getText().toString(),
            editPass.getText().toString());
            break;
    }
}

@Override
public void onClearText() {
    editUser.setText("");
    editPass.setText("");
}

@Override
public void onLoginResult(Boolean result, int code) {
    loginPresenter.setProgressBarVisibility(View.INVISIBLE);
    btnLogin.setEnabled(true);
    btnClear.setEnabled(true);
    if (result){
        Toast.makeText(this,"登录成功",Toast.LENGTH_SHORT).show();
    } else
        Toast.makeText(this,"登录失败，代码 = " + code,Toast.LENGTH_SHORT).show();
}

@Override
protected void onDestroy() {
    super.onDestroy();
}

@Override
public void onSetProgressBarVisibility(int visibility) {
    progressBar.setVisibility(visibility);
}
}

```

创建一个 ILoginView 接口

在 view 文件夹下创建一个 ILoginView 接口，用于从 Presenter 更新信息，内容如下：

```

public interface ILoginView {
    public void onClearText();
    public void onLoginResult(Boolean result, int code);
    public void onSetProgressBarVisibility(int visibility);
}

```

创建一个 ILoginPresenter 接口

创建一个 ILoginPresenter 接口，用于与 LoginActivity (视图) 通信，并创建 LoginPresenterCompl 类来处理登录功能并向 Activity 报告。该 LoginPresenterCompl 类实现了 ILoginPresenter 接口：

ILoginPresenter.class

```
public interface ILoginPresenter {
```

```

public void onClick(View v) {
    switch (v.getId()) {
        case R.id.btn_login_clear:
            loginPresenter.clear();
            break;
        case R.id.btn_login_login:
            loginPresenter.setProgressBarVisibility(View.VISIBLE);
            btnLogin.setEnabled(false);
            btnClear.setEnabled(false);
            loginPresenter.doLogin(editUser.getText().toString(),
            editPass.getText().toString());
            break;
    }
}

@Override
public void onClearText() {
    editUser.setText("");
    editPass.setText("");
}

@Override
public void onLoginResult(Boolean result, int code) {
    loginPresenter.setProgressBarVisibility(View.INVISIBLE);
    btnLogin.setEnabled(true);
    btnClear.setEnabled(true);
    if (result){
        Toast.makeText(this,"Login Success",Toast.LENGTH_SHORT).show();
    } else
        Toast.makeText(this,"Login Fail, code = " + code,Toast.LENGTH_SHORT).show();
}

```

Creating an ILoginView Interface

Create an ILoginView interface for update info from Presenter under view folder as follows:

```

public interface ILoginView {
    public void onClearText();
    public void onLoginResult(Boolean result, int code);
    public void onSetProgressBarVisibility(int visibility);
}

```

Creating an ILoginPresenter Interface

Create an ILoginPresenter interface in order to communicate with LoginActivity (Views) and create the LoginPresenterCompl class for handling login functionality and reporting back to the Activity. The LoginPresenterCompl class implements the ILoginPresenter interface:

ILoginPresenter.class

```
public interface ILoginPresenter {
```

```

void clear();
void doLogin(String name, String passwd);
void setProgressBarVisibility(int visibility);
}

```

LoginPresenterCompl.class

```

public class LoginPresenterCompl implements ILoginPresenter {
    ILoginView iLoginView;
    IUser user;
    Handler handler;

    public LoginPresenterCompl(ILoginView iLoginView) {
        this.iLoginView = iLoginView;
        initUser();
        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public void clear() {
        iLoginView.onClearText();
    }

    @Override
    public void doLogin(String name, String passwd) {
        Boolean isLoginSuccess = true;
        final int code = user.checkUserValidity(name,passwd);
        if (code!=0) isLoginSuccess = false;
        final Boolean result = isLoginSuccess;
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                iLoginView.onLoginResult(result, code);
            }
        }, 5000);
    }

    @Override
    public void setProgressBarVisibility(int visibility){
        iLoginView.onSetProgressBarVisibility(visibility);
    }

    private void initUser(){
        user = new UserModel("mvp","mvp");
    }
}

```

创建一个 UserModel

创建一个 UserModel，它类似于 LoginActivity 的 Pojo 类。为 Pojo 验证创建一个 IUser 接口：

UserModel.class

```

public class UserModel implements IUser {
    String name;
    String passwd;

    public UserModel(String name, String passwd) {
        this.name = name;
        this.passwd = passwd;
    }

    @Override

```

```

void clear();
void doLogin(String name, String passwd);
void setProgressBarVisibility(int visibility);
}

```

LoginPresenterCompl.class

```

public class LoginPresenterCompl implements ILoginPresenter {
    ILoginView iLoginView;
    IUser user;
    Handler handler;

    public LoginPresenterCompl(ILoginView iLoginView) {
        this.iLoginView = iLoginView;
        initUser();
        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public void clear() {
        iLoginView.onClearText();
    }

    @Override
    public void doLogin(String name, String passwd) {
        Boolean isLoginSuccess = true;
        final int code = user.checkUserValidity(name,passwd);
        if (code!=0) isLoginSuccess = false;
        final Boolean result = isLoginSuccess;
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                iLoginView.onLoginResult(result, code);
            }
        }, 5000);
    }

    @Override
    public void setProgressBarVisibility(int visibility){
        iLoginView.onSetProgressBarVisibility(visibility);
    }

    private void initUser(){
        user = new UserModel("mvp","mvp");
    }
}

```

Creating a UserModel

Create a UserModel which is like a Pojo class for LoginActivity. Create an IUser interface for Pojo validations:

UserModel.class

```

public class UserModel implements IUser {
    String name;
    String passwd;

    public UserModel(String name, String passwd) {
        this.name = name;
        this.passwd = passwd;
    }

    @Override

```

```

public String getName() {
    return name;
}

@Override
public String getPasswd() {
    return passwd;
}

@Override
public int checkUserValidity(String name, String passwd){
    if (name==null||passwd==null||!name.equals(getName())||!passwd.equals(getPasswd())){
        return -1;
    }
    return 0;
}

```

IUser.class

```

public interface IUser {
    String getName();

    String getPasswd();

    int checkUserValidity(String name, String passwd);
}

```

MVP

模型-视图-主人（MVP）是模型-视图-控制器（MVC）架构模式的一个派生。它主要用于构建用户界面，并提供以下优势：

- 视图与模型更加分离。主人是模型与视图之间的中介。
- 更容易创建单元测试。
- 通常，视图与主人之间是一对一的映射，对于复杂视图，可以使用多个主人。

```

public String getName() {
    return name;
}

@Override
public String getPasswd() {
    return passwd;
}

@Override
public int checkUserValidity(String name, String passwd){
    if (name==null||passwd==null||!name.equals(getName())||!passwd.equals(getPasswd())){
        return -1;
    }
    return 0;
}

```

IUser.class

```

public interface IUser {
    String getName();

    String getPasswd();

    int checkUserValidity(String name, String passwd);
}

```

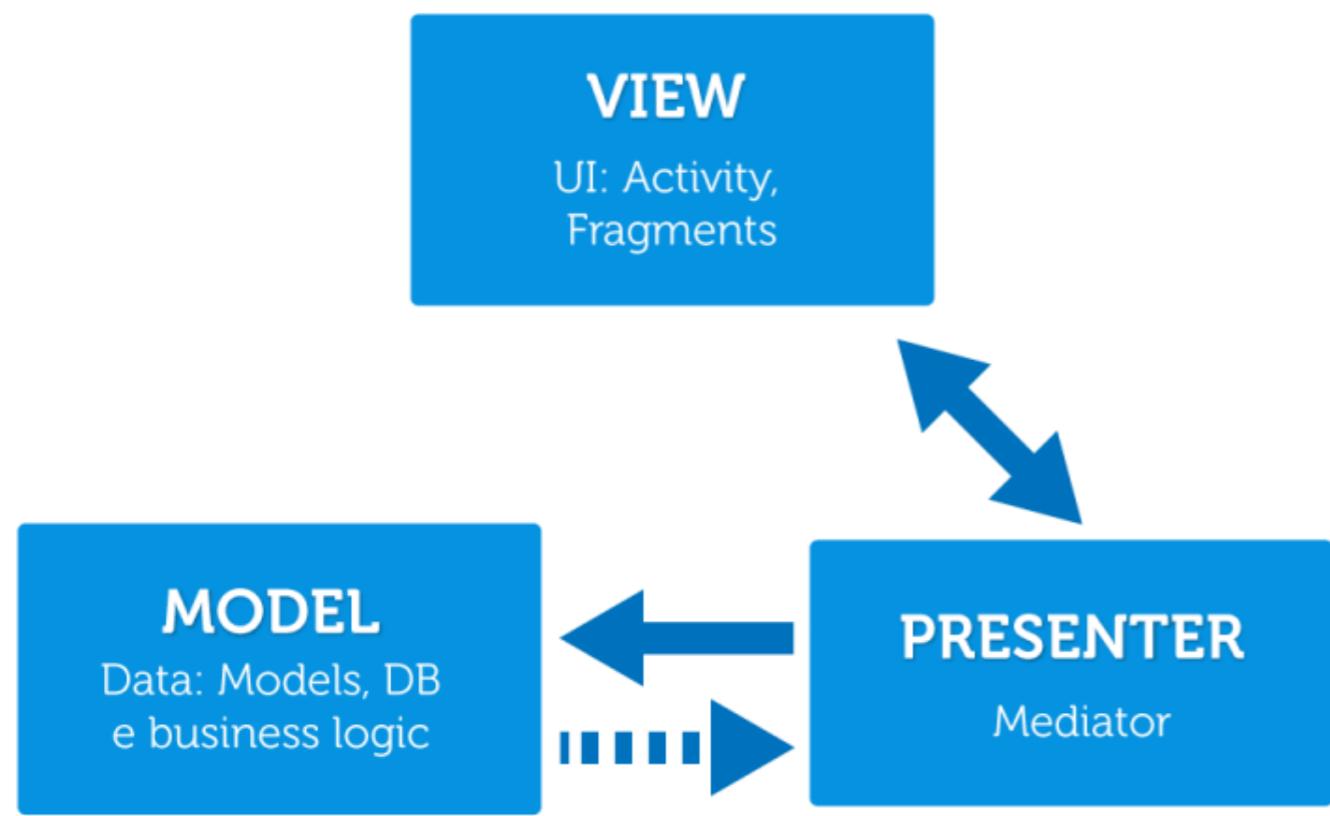
MVP

A Model-view-presenter (MVP) is a derivation of the model-view-controller (MVC) architectural pattern. It is used mostly for building user interfaces and offers the following benefits:

- Views are more separated from Models. The Presenter is the mediator between Model and View.
- It is easier to create unit tests.
- Generally, there is a one-to-one mapping between View and Presenter, with the possibility to use multiple Presenters for complex Views.

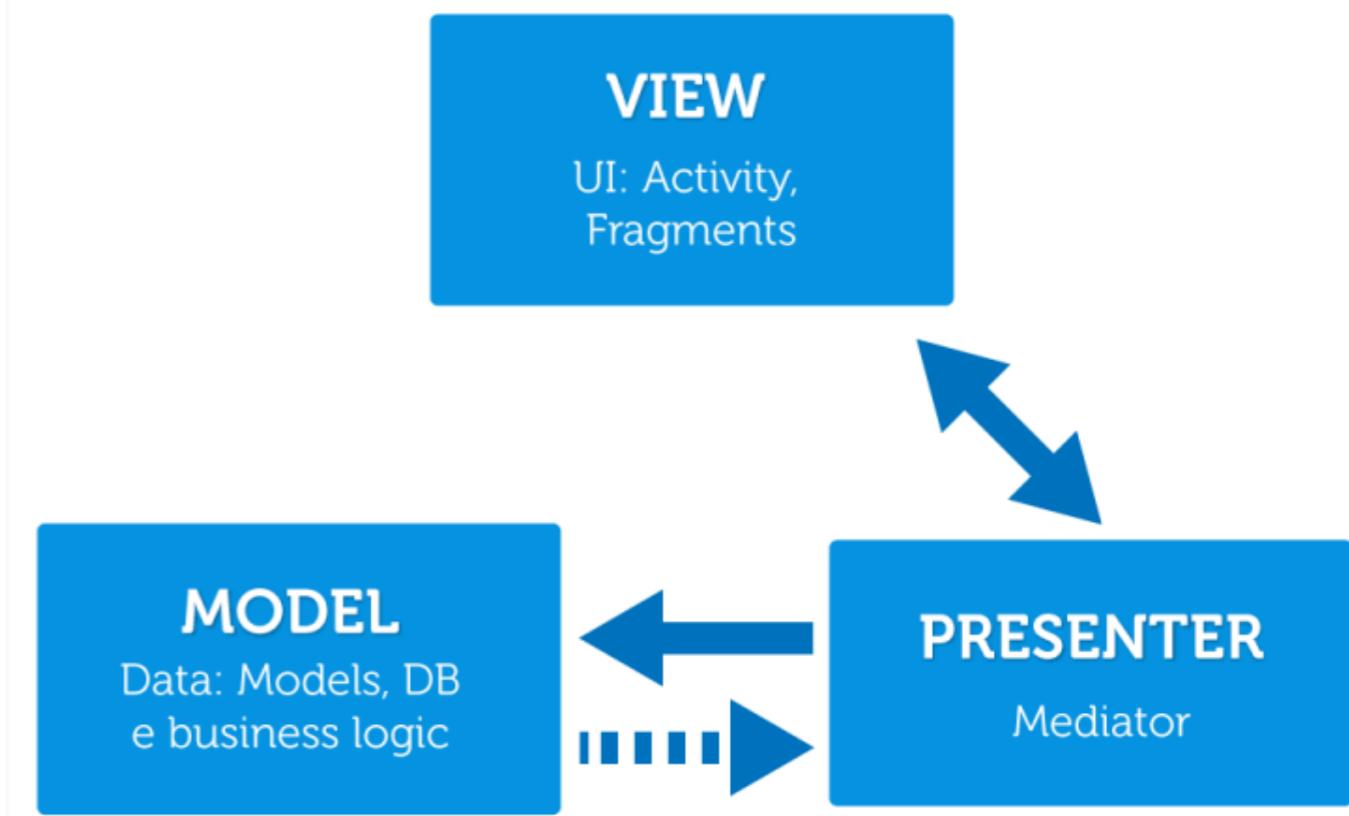
MVP

Model View Presenter



MVP

Model View Presenter



第135章：方向变化

第135.1节：保存和恢复活动状态

当您的活动开始停止时，系统会调用`onSaveInstanceState()`，以便您的活动可以通过一组键值对保存状态信息。该方法的默认实现会自动保存有关活动视图层次结构状态的信息，例如EditText控件中的文本或滚动位置。

[ListView](#).

为了保存活动的额外状态信息，必须实现`onSaveInstanceState()`，并向Bundle对象添加键值对。例如：

```
public class MainActivity extends Activity {
    static final String SOME_VALUE = "int_value";
    static final String SOME_OTHER_VALUE = "string_value";

    @Override
    protected void onSaveInstanceState(Bundle savedInstanceState) {
        // 将自定义值保存到bundle中
        savedInstanceState.putInt(SOME_VALUE, someIntValue);
        savedInstanceState.putString(SOME_OTHER_VALUE, someStringValue);
        // 始终调用超类方法以保存视图层次结构状态
        super.onSaveInstanceState(savedInstanceState);
    }
}
```

系统会在Activity销毁之前调用该方法。之后系统会调用`onRestoreInstanceState`，我们可以从bundle中恢复状态：

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    // 始终调用超类方法以恢复视图层次结构
    super.onRestoreInstanceState(savedInstanceState);
    // 从保存的实例恢复状态成员
    someIntValue = savedInstanceState.getInt(SOME_VALUE);
    someStringValue = savedInstanceState.getString(SOME_OTHER_VALUE);
}
```

实例状态也可以在标准的`Activity#onCreate`方法中恢复，但在`onRestoreInstanceState`中恢复更为方便，因为它确保所有初始化工作已完成，并允许子类决定是否使用默认实现。详情请阅读此[stackoverflow帖子](#)。

注意，`onSaveInstanceState` 和 `onRestoreInstanceState` 并不保证会一起被调用。Android 会在活动可能被销毁时调用`onSaveInstanceState()`。然而，有些情况下虽然调用了`onSaveInstanceState`，但活动并未被销毁，因此`onRestoreInstanceState`不会被调用。

第135.2节：保留Fragment

在许多情况下，通过简单使用Fragment可以避免活动重新创建时出现的问题。如果你的视图和状态都在Fragment中，我们可以轻松地在活动重新创建时保留该Fragment：

```
public class RetainedFragment extends Fragment {
    // 我们想要保留的数据对象
    private MyDataObject data;
```

Chapter 135: Orientation Changes

Section 135.1: Saving and Restoring Activity State

As your activity begins to stop, the system calls`onSaveInstanceState()` so your activity can save state information with a collection of key-value pairs. The default implementation of this method automatically saves information about the state of the activity's view hierarchy, such as the text in an EditText widget or the scroll position of a [ListView](#).

To save additional state information for your activity, you must implement`onSaveInstanceState()` and add key-value pairs to the Bundle object. For example:

```
public class MainActivity extends Activity {
    static final String SOME_VALUE = "int_value";
    static final String SOME_OTHER_VALUE = "string_value";

    @Override
    protected void onSaveInstanceState(Bundle savedInstanceState) {
        // Save custom values into the bundle
        savedInstanceState.putInt(SOME_VALUE, someIntValue);
        savedInstanceState.putString(SOME_OTHER_VALUE, someStringValue);
        // Always call the superclass so it can save the view hierarchy state
        super.onSaveInstanceState(savedInstanceState);
    }
}
```

The system will call that method before an Activity is destroyed. Then later the system will call`onRestoreInstanceState` where we can restore state from the bundle:

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);
    // Restore state members from saved instance
    someIntValue = savedInstanceState.getInt(SOME_VALUE);
    someStringValue = savedInstanceState.getString(SOME_OTHER_VALUE);
}
```

Instance state can also be restored in the standard`Activity#onCreate` method but it is convenient to do it in`onRestoreInstanceState` which ensures all of the initialization has been done and allows subclasses to decide whether to use the default implementation. Read this [stackoverflow post](#) for details.

Note that`onSaveInstanceState` and`onRestoreInstanceState` are not guaranteed to be called together. Android invokes`onSaveInstanceState()` when there's a chance the activity might be destroyed. However, there are cases where`onSaveInstanceState` is called but the activity is not destroyed and as a result`onRestoreInstanceState` is not invoked.

Section 135.2: Retaining Fragments

In many cases, we can avoid problems when an Activity is re-created by simply using fragments. If your views and state are within a fragment, we can easily have the fragment be retained when the activity is re-created:

```
public class RetainedFragment extends Fragment {
    // data object we want to retain
    private MyDataObject data;
```

```
// 此方法仅在该片段中调用一次
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 在活动重新初始化时保留此片段
    setRetainInstance(true);
}

public void setData(MyDataObject data) {
    this.data = data;
}

public MyDataObject getData() {
    return data;
}
}
```

这种方法可以防止片段在活动生命周期内被销毁。它们会被保留在片段管理器中。更多信息请参见Android官方文档。

现在你可以在创建片段之前通过标签检查片段是否已存在，且片段将在配置更改时保留其状态。更多详情请参见“处理运行时更改”指南。

第135.3节：手动管理配置更改

如果你的应用在特定配置更改期间不需要更新资源，且存在性能限制需要避免活动重启，那么你可以声明你的活动自行处理配置更改，这样系统就不会重启你的活动。

然而，这种技术应视为最后手段，仅在必须避免因配置更改导致重启时使用，不推荐大多数应用采用。要采取此方法，我们必须在AndroidManifest.xml中的活动节点添加android:configChanges属性：

```
<activity android:name=".MyActivity"
    android:configChanges="orientation|screenSize|keyboardHidden"
    android:label="@string/app_name">
```

现在，当这些配置之一发生变化时，活动不会重新启动，而是会收到对onConfigurationChanged()：

```
// 在接收这些变化的活动中
// 检查当前设备方向，并相应地显示提示
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // 检查屏幕方向
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "横屏", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "竖屏", Toast.LENGTH_SHORT).show();
    }
}
```

请参阅“处理配置变化”文档。有关您可以在活动中处理的配置变化的更多信息，请参阅android:configChanges文档和Configuration类。

```
// this method is only called once for this fragment
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // retain this fragment when activity is re-initialized
    setRetainInstance(true);
}

public void setData(MyDataObject data) {
    this.data = data;
}

public MyDataObject getData() {
    return data;
}
}
```

This approach keeps the fragment from being destroyed during the activity lifecycle. They are instead retained inside the Fragment Manager. See the Android official docs for more [information](#).

Now you can check to see if the fragment already exists by tag before creating one and the fragment will retain its state across configuration changes. See the Handling Runtime Changes guide for [more details](#).

Section 135.3: Manually Managing Configuration Changes

If your application doesn't need to update resources during a specific configuration change and you have a performance limitation that requires you to avoid the activity restart, then you can declare that your activity handles the configuration change itself, which prevents the system from restarting your activity.

However, this technique should be considered a last resort when you must avoid restarts due to a configuration change and is not recommended for most applications. To take this approach, we must add the android:configChanges node to the activity within the [AndroidManifest.xml](#):

```
<activity android:name=".MyActivity"
    android:configChanges="orientation|screenSize|keyboardHidden"
    android:label="@string/app_name">
```

Now, when one of these configurations change, the activity does not restart but instead receives a call to onConfigurationChanged():

```
// Within the activity which receives these changes
// Checks the current device orientation, and toasts accordingly
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

See the [Handling the Change](#) docs. For more about which configuration changes you can handle in your activity, see the [android:configChanges](#) documentation and the [Configuration](#) class.

第135.4节：处理AsyncTask

问题：

- 如果在AsyncTask启动后发生屏幕旋转，所属的活动会被销毁并重新创建。
- 当AsyncTask完成时，它想要更新可能已经无效的用户界面。

解决方案：

使用Loaders，可以轻松克服活动的销毁/重建问题。

示例：

MainActivity：

```
public class MainActivity extends AppCompatActivity
    implements LoaderManager.LoaderCallbacks<Bitmap> {

    //加载器的唯一ID
    private static final int MY_LOADER = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        LoaderManager loaderManager = getSupportFragmentManager();

        if(loaderManager.getLoader(MY_LOADER) == null) {
            loaderManager.initLoader(MY_LOADER, null, this).forceLoad();
        }
    }

    @Override
    public Loader<Bitmap> onCreateLoader(int id, Bundle args) {
        //创建你的Loader<Bitmap>的新实例
        MyLoader loader = new MyLoader(MainActivity.this);
        return loader;
    }

    @Override
    public void onLoadFinished(Loader<Bitmap> loader, Bitmap data) {
        // 在父活动/服务中执行某些操作
        // 即显示下载的图片
        Log.d("MyAsyncTask", "Received result: ");
    }

    @Override
    public void onLoaderReset(Loader<Bitmap> loader) {
    }
}
```

AsyncTaskLoader：

```
public class MyLoader extends AsyncTaskLoader<Bitmap> {
    private WeakReference<Activity> motherActivity;

    public MyLoader(Activity activity) {
        super(activity);
        //我们不使用这个，但如果你想用可以用，不过记住，WeakReference
        motherActivity = new WeakReference<>(activity);
```

Section 135.4: Handling AsyncTask

Problem:

- If after the AsyncTask starts there is a screen rotation the owning activity is destroyed and recreated.
- When the AsyncTask finishes it wants to update the UI that may not valid anymore.

Solution:

Using Loaders, one can easily overcome the activity destruction/recreation.

Example:

MainActivity:

```
public class MainActivity extends AppCompatActivity
    implements LoaderManager.LoaderCallbacks<Bitmap> {

    //Unique id for the loader
    private static final int MY_LOADER = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        LoaderManager loaderManager = getSupportFragmentManager();

        if(loaderManager.getLoader(MY_LOADER) == null) {
            loaderManager.initLoader(MY_LOADER, null, this).forceLoad();
        }
    }

    @Override
    public Loader<Bitmap> onCreateLoader(int id, Bundle args) {
        //Create a new instance of your Loader<Bitmap>
        MyLoader loader = new MyLoader(MainActivity.this);
        return loader;
    }

    @Override
    public void onLoadFinished(Loader<Bitmap> loader, Bitmap data) {
        // do something in the parent activity/service
        // i.e. display the downloaded image
        Log.d("MyAsyncTask", "Received result: ");
    }

    @Override
    public void onLoaderReset(Loader<Bitmap> loader) {
    }
}
```

AsyncTaskLoader：

```
public class MyLoader extends AsyncTaskLoader<Bitmap> {
    private WeakReference<Activity> motherActivity;

    public MyLoader(Activity activity) {
        super(activity);
        //We don't use this, but if you want you can use it, but remember, WeakReference
        motherActivity = new WeakReference<>(activity);
```

```

}
@Override
public Bitmap loadInBackground() {
    // 执行工作。例如从互联网下载一张图片以在界面中显示。
    // 即返回下载的界面图片
    return result;
}
}

```

注意：

重要的是要么使用v4兼容库，要么不使用，但不要部分使用一个部分使用另一个，因为这会导致编译错误。可以通过查看导入的`android.support.v4.content`和`android.content`（不应同时存在）来检查。

第135.5节：以编程方式锁定锁屏的旋转

在开发过程中，通常会发现在代码的特定部分锁定/解锁设备屏幕非常有用。

例如，在显示带有信息的对话框时，开发者可能想要锁定屏幕旋转，以防止对话框被关闭并且当前活动被重建，在对话框关闭时再解锁屏幕旋转。

尽管我们可以通过在清单文件中这样做来实现旋转锁定：

```

<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>

```

也可以通过编程方式实现，方法如下：

```

public void lockDeviceRotation(boolean value) {
    if (value) {
        int currentOrientation = getResources().getConfiguration().orientation;
        if (currentOrientation == Configuration.ORIENTATION_LANDSCAPE) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT);
        }
    } else {
        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_USER);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
        }
    }
}

```

然后调用以下方法，分别锁定和解锁设备旋转

`lockDeviceRotation(true)`

和

```

}
@Override
public Bitmap loadInBackground() {
    // Do work. I.e download an image from internet to be displayed in gui.
    // i.e. return the downloaded gui
    return result;
}
}

```

Note:

It is important to use either the v4 compatibility library or not, but do not use part of one and part of the other, as it will lead to compilation errors. To check you can look at the imports for `android.support.v4.content` and `android.content` (you shouldn't have both).

Section 135.5: Lock Screen's rotation programmatically

It is very common that during development, one may find **very useful to lock/unlock the device screen during specific parts of the code.**

For instance, while showing a Dialog with information, the developer might want to **lock** the screen's rotation to prevent the dialog from being dismissed and the current activity from being rebuilt to **unlock** it again when the dialog is dismissed.

Even though we can achieve rotation locking from the manifest by doing :

```

<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>

```

One can do it programmatically as well by doing the following :

```

public void lockDeviceRotation(boolean value) {
    if (value) {
        int currentOrientation = getResources().getConfiguration().orientation;
        if (currentOrientation == Configuration.ORIENTATION_LANDSCAPE) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT);
        }
    } else {
        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_USER);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
        }
    }
}

```

And then calling the following, to respectively lock and unlock the device rotation

`lockDeviceRotation(true)`

and

```
lockDeviceRotation(false)
```

第135.6节：保存和恢复Fragment状态

Fragment还有一个onSaveInstanceState()方法，当需要保存其状态时会调用：

```
public class MySimpleFragment extends Fragment {
    private int someStateValue;
    private final String SOME_VALUE_KEY = "someValueToSave";

    // 当配置发生变化且Fragment需要保存状态时触发
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putInt(SOME_VALUE_KEY, someStateValue);
        super.onSaveInstanceState(outState);
    }
}
```

然后我们可以在onCreateView中从这个保存的状态中提取数据：

```
public class MySimpleFragment extends Fragment {
    // ...

    // 根据布局XML为Fragment加载视图
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.my_simple_fragment, container, false);
        if (savedInstanceState != null) {
            someStateValue = savedInstanceState.getInt(SOME_VALUE_KEY);
            // 如有需要，对值进行处理
        }
        return view;
    }
}
```

为了正确保存片段状态，我们需要确保在配置更改时不会不必要的重新创建片段。这意味着要小心不要在片段已经存在时重新初始化它们。

在活动中初始化的任何片段，在配置更改后都需要通过标签查找：

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) { // 有保存的实例状态，片段可能已存在
            // 通过标签查找已存在的实例
            fragmentSimple = (MySimpleFragment)
                getSupportFragmentManager().findFragmentByTag(SIMPLE_FRAGMENT_TAG);
        } else if (fragmentSimple == null) {
            // 仅在片段尚未实例化时创建片段
            fragmentSimple = new MySimpleFragment();
        }
    }
}
```

这要求我们在将片段放入事务中的活动中，务必小心地包含一个用于查找的标签：

```
lockDeviceRotation(false)
```

Section 135.6: Saving and Restoring Fragment State

Fragments also have a onSaveInstanceState() method which is called when their state needs to be saved:

```
public class MySimpleFragment extends Fragment {
    private int someStateValue;
    private final String SOME_VALUE_KEY = "someValueToSave";

    // Fires when a configuration change occurs and fragment needs to save state
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putInt(SOME_VALUE_KEY, someStateValue);
        super.onSaveInstanceState(outState);
    }
}
```

Then we can pull data out of this saved state in onCreateView:

```
public class MySimpleFragment extends Fragment {
    // ...

    // Inflate the view for the fragment based on layout XML
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.my_simple_fragment, container, false);
        if (savedInstanceState != null) {
            someStateValue = savedInstanceState.getInt(SOME_VALUE_KEY);
            // Do something with value if needed
        }
        return view;
    }
}
```

For the fragment state to be saved properly, we need to be sure that we aren't unnecessarily recreating the fragment on configuration changes. This means being careful not to reinitialize existing fragments when they already exist. Any fragments being initialized in an Activity need to be looked up by tag after a configuration change:

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) { // saved instance state, fragment may exist
            // look up the instance that already exists by tag
            fragmentSimple = (MySimpleFragment)
                getSupportFragmentManager().findFragmentByTag(SIMPLE_FRAGMENT_TAG);
        } else if (fragmentSimple == null) {
            // only create fragment if they haven't been instantiated already
            fragmentSimple = new MySimpleFragment();
        }
    }
}
```

This requires us to be careful to include a tag for lookup whenever putting a fragment into the activity within a

事务：

```
public class ParentActivity extends AppCompatActivity {  
    private MySimpleFragment fragmentSimple;  
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        // ... 上面进行片段查找或实例化 ...  
        // 始终为插入到容器中的片段添加标签  
        if (!fragmentSimple.isInLayout()) {  
            getSupportFragmentManager()  
                .beginTransaction()  
                .replace(R.id.container, fragmentSimple, SIMPLE_FRAGMENT_TAG)  
                .commit();  
        }  
    }  
}
```

通过这个简单的模式，我们可以正确地重用片段并在配置更改时恢复它们的状态。

transaction:

```
public class ParentActivity extends AppCompatActivity {  
    private MySimpleFragment fragmentSimple;  
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        // ... fragment lookup or instantiation from above...  
        // Always add a tag to a fragment being inserted into container  
        if (!fragmentSimple.isInLayout()) {  
            getSupportFragmentManager()  
                .beginTransaction()  
                .replace(R.id.container, fragmentSimple, SIMPLE_FRAGMENT_TAG)  
                .commit();  
        }  
    }  
}
```

With this simple pattern, we can properly re-use fragments and restore their state across configuration changes.

第136章：Xposed

第136.1节：创建Xposed模块

Xposed是一个允许你钩取其他应用方法调用的框架。当你通过反编译APK进行修改时，可以直接在任何位置插入或更改命令。然而，你需要随后重新编译/签名APK，并且只能分发整个包。使用Xposed，你可以在方法执行前后注入自己的代码，或者完全替换整个方法。不幸的是，Xposed只能安装在已root的设备上。每当你想操控其他应用或核心Android系统的行为，并且不想经历反编译、重新编译和签名APK的麻烦时，都应该使用Xposed。

首先，你需要在Android Studio中创建一个没有Activity的标准应用。

然后你必须在你的build.gradle中包含以下代码：

```
repositories {  
    jcenter();  
}
```

之后你添加以下依赖项：

```
provided 'de.robv.android.xposed:api:82'  
provided 'de.robv.android.xposed:api:82:sources'
```

现在你必须将这些标签放置在AndroidManifest.xml中的application标签内，以便Xposed识别你的模块：

```
<meta-data  
    android:name="xposedmodule"  
    android:value="true" />  
<meta-data  
    android:name="xposeddescription"  
    android:value="YOUR_MODULE_DESCRIPTION" />  
<meta-data  
    android:name="xposedminversion"  
    android:value="82" />
```

注意：始终将82替换为最新的Xposed版本。

第136.2节：钩取方法

创建一个实现IXposedHookLoadPackage的新类，并实现handleLoadPackage方法：

```
public class MultiPatcher implements IXposedHookLoadPackage  
{  
    @Override  
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws  
Throwable  
    {  
    }  
}
```

在该方法内部，检查loadPackageParam.packageName以获取你想钩取的应用包名：

Chapter 136: Xposed

Section 136.1: Creating a Xposed Module

Xposed is a framework that allows you to hook method calls of other apps. When you do a modification by decompiling an APK, you can insert/change commands directly wherever you want. However, you will need to recompile/sign the APK afterwards and you can only distribute the whole package. With Xposed, you can inject your own code before or after methods, or replace whole methods completely. Unfortunately, you can only install Xposed on rooted devices. You should use Xposed whenever you want to manipulate the behavior of other apps or the core Android system and don't want to go through the hassle of decompiling, recompiling and signing APKs.

First, you create a standard app without an Activity in Android Studio.

Then you have to include the following code in your build.gradle:

```
repositories {  
    jcenter();  
}
```

After that you add the following dependencies:

```
provided 'de.robv.android.xposed:api:82'  
provided 'de.robv.android.xposed:api:82:sources'
```

Now you have to place these tags inside the application tag found in the AndroidManifest.xml so Xposed recognizes your module:

```
<meta-data  
    android:name="xposedmodule"  
    android:value="true" />  
<meta-data  
    android:name="xposeddescription"  
    android:value="YOUR_MODULE_DESCRIPTION" />  
<meta-data  
    android:name="xposedminversion"  
    android:value="82" />
```

NOTE: Always replace 82 with the [latest Xposed version](#).

Section 136.2: Hooking a method

Create a new class implementing IXposedHookLoadPackage and implement the handleLoadPackage method:

```
public class MultiPatcher implements IXposedHookLoadPackage  
{  
    @Override  
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws  
Throwable  
    {  
    }  
}
```

Inside the method, you check loadPackageParam.packageName for the package name of the app you want to hook:

```

@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}

```

现在你可以钩住你的方法，并在其代码运行之前或之后进行操作：

```

@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }

XposedHelpers.findAndHookMethod(
    "other.package.name",
    loadPackageParam.classLoader,
    "otherMethodName",
    YourFirstParameter.class,
    YourSecondParameter.class,
    new XC_MethodHook()
    {
        @Override
        protected void beforeHookedMethod(MethodHookParam param) throws Throwable
        {
            Object[] args = param.args;

args[0] = true;
args[1] = "example string";
args[2] = 1;

Object thisObject = param.thisObject;

// 对该类的实例进行操作
        }

        @Override
        protected void afterHookedMethod(MethodHookParam param) throws Throwable
        {
            Object result = param.getResult();

param.setResult(result + "example string");
        }
    });
}

```

```

@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}

```

Now you can hook your method and either manipulate it before its code is run, or after:

```

@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }

XposedHelpers.findAndHookMethod(
    "other.package.name",
    loadPackageParam.classLoader,
    "otherMethodName",
    YourFirstParameter.class,
    YourSecondParameter.class,
    new XC_MethodHook()
    {
        @Override
        protected void beforeHookedMethod(MethodHookParam param) throws Throwable
        {
            Object[] args = param.args;

args[0] = true;
args[1] = "example string";
args[2] = 1;

Object thisObject = param.thisObject;

// Do something with the instance of the class
        }

        @Override
        protected void afterHookedMethod(MethodHookParam param) throws Throwable
        {
            Object result = param.getResult();

param.setResult(result + "example string");
        }
    });
}

```

第137章：包管理器

第137.1节：获取应用版本

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
        getApplicationContext().getPackageName(),
        0);

    return info.versionName;
}
```

第137.2节：版本名称和版本代码

要获取应用当前构建的versionName和versionCode，应查询Android的包管理器。

```
try {
    // 引用 Android 的包管理器
    PackageManager packageManager = this.getPackageManager();

    // 获取此应用的包信息
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // 版本代码
    info.versionCode

    // 版本名称
    info.versionName

} catch (NameNotFoundException e) {
    // 处理异常
}
```

第137.3节：安装时间和更新时间

要获取应用的安装时间或更新时间，应查询 Android 的包管理器。

```
try {
    // 引用 Android 的包管理器
    PackageManager packageManager = this.getPackageManager();

    // 获取此应用的包信息
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // 安装时间。单位与 currentTimeMillis() 相同。
    info.firstInstallTime

    // 最后更新时间。单位与 currentTimeMillis() 相同。
    info.lastUpdateTime

} catch (NameNotFoundException e) {
    // 处理异常
}
```

Chapter 137: PackageManager

Section 137.1: Retrieve application version

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
        getApplicationContext().getPackageName(),
        0);

    return info.versionName;
}
```

Section 137.2: Version name and version code

To get versionName and versionCode of current build of your application you should query Android's package manager.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Version code
    info.versionCode

    // Version name
    info.versionName

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Section 137.3: Install time and update time

To get the time at which your app was installed or updated, you should query Android's package manager.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Install time. Units are as per currentTimeMillis().
    info.firstInstallTime

    // Last update time. Units are as per currentTimeMillis().
    info.lastUpdateTime

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

第137.4节：使用 PackageManager 的实用方法

这里我们可以找到一些使用 PackageManager 的有用方法，

下面的方法将帮助通过包名获取应用名称

```
private String getAppNameFromPackage(String packageName, Context context) {
    Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
    mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> pkgAppsList = context.getPackageManager()
        .queryIntentActivities(mainIntent, 0);
    for (ResolveInfo app : pkgAppsList) {
        if (app.activityInfo.packageName.equals(packageName)) {
            return app.activityInfo.loadLabel(context.getPackageManager()).toString();
        }
    }
    return null;
}
```

以下方法将帮助通过包名获取应用图标，

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

以下方法将帮助获取已安装应用列表。

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {
    List<ApplicationInfo> list =
    packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

注意：上述方法也会返回启动器应用。

以下方法将帮助隐藏启动器中的应用图标。

```
public static void hideLockerApp(Context context, boolean hide) {
    ComponentName componentName = new ComponentName(context.getApplicationContext(),
        SplashActivity.class);

    int setting = hide ? PackageManager.COMPONENT_ENABLED_STATE_DISABLED
        : PackageManager.COMPONENT_ENABLED_STATE_ENABLED;

    int current = context.getPackageManager().getComponentEnabledSetting(componentName);

    if (current != setting) {
        context.getPackageManager().setComponentEnabledSetting(componentName, setting,
            PackageManager.DONT_KILL_APP);
    }
}
```

Section 137.4: Utility method using PackageManager

Here we can find some useful method using PackageManager,

Below method will help to get the app name using package name

```
private String getAppNameFromPackage(String packageName, Context context) {
    Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
    mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> pkgAppsList = context.getPackageManager()
        .queryIntentActivities(mainIntent, 0);
    for (ResolveInfo app : pkgAppsList) {
        if (app.activityInfo.packageName.equals(packageName)) {
            return app.activityInfo.loadLabel(context.getPackageManager()).toString();
        }
    }
    return null;
}
```

Below method will help to get the app icon using package name,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

Below method will help to get the list of installed application.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {
    List<ApplicationInfo> list =
    packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Note: above method will give the launcher application too.

Below method will help to hide the app icon from the launcher.

```
public static void hideLockerApp(Context context, boolean hide) {
    ComponentName componentName = new ComponentName(context.getApplicationContext(),
        SplashActivity.class);

    int setting = hide ? PackageManager.COMPONENT_ENABLED_STATE_DISABLED
        : PackageManager.COMPONENT_ENABLED_STATE_ENABLED;

    int current = context.getPackageManager().getComponentEnabledSetting(componentName);

    if (current != setting) {
        context.getPackageManager().setComponentEnabledSetting(componentName, setting,
            PackageManager.DONT_KILL_APP);
    }
}
```

}

注意：关闭设备后再开机，此图标将在启动器中恢复显示。

}

Note: After switch off the device and switch on this icon will come back in the launcher.

第138章：手势检测

第138.1节：滑动检测

```
public class OnSwipeListener implements View.OnTouchListener {  
  
    private final GestureDetector gestureDetector;  
  
    public OnSwipeListener(Context context) {  
        gestureDetector = new GestureDetector(context, new GestureListener());  
    }  
  
    @Override  
    public boolean onTouch(View v, MotionEvent event) {  
        return gestureDetector.onTouchEvent(event);  
    }  
  
    private final class GestureListener extends GestureDetector.SimpleOnGestureListener {  
  
        private static final int SWIPE_VELOCITY_THRESHOLD = 100;  
        private static final int SWIPE_THRESHOLD = 100;  
  
        @Override  
        public boolean onDown(MotionEvent e) {  
            return true;  
        }  
  
        @Override  
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {  
            float diffY = e2.getY() - e1.getY();  
            float diffX = e2.getX() - e1.getX();  
            if (Math.abs(diffX) > Math.abs(diffY)) {  
                if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >  
                    SWIPE_VELOCITY_THRESHOLD) {  
                    if (diffX > 0) {  
                        onSwipeRight();  
                    } 否则 {  
                        onSwipeLeft();  
                    }  
                }  
            } 否则如果 (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) >  
                SWIPE_VELOCITY_THRESHOLD) {  
                如果 (diffY > 0) {  
                    onSwipeBottom();  
                } 否则 {  
                    onSwipeTop();  
                }  
            }  
            return true;  
        }  
  
        public void onSwipeRight() {}  
  
        public void onSwipeLeft() {}  
  
        public void onSwipeTop() {}  
    }  
}
```

Chapter 138: Gesture Detection

Section 138.1: Swipe Detection

```
public class OnSwipeListener implements View.OnTouchListener {  
  
    private final GestureDetector gestureDetector;  
  
    public OnSwipeListener(Context context) {  
        gestureDetector = new GestureDetector(context, new GestureListener());  
    }  
  
    @Override  
    public boolean onTouch(View v, MotionEvent event) {  
        return gestureDetector.onTouchEvent(event);  
    }  
  
    private final class GestureListener extends GestureDetector.SimpleOnGestureListener {  
  
        private static final int SWIPE_VELOCITY_THRESHOLD = 100;  
        private static final int SWIPE_THRESHOLD = 100;  
  
        @Override  
        public boolean onDown(MotionEvent e) {  
            return true;  
        }  
  
        @Override  
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {  
            float diffY = e2.getY() - e1.getY();  
            float diffX = e2.getX() - e1.getX();  
            if (Math.abs(diffX) > Math.abs(diffY)) {  
                if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >  
                    SWIPE_VELOCITY_THRESHOLD) {  
                    if (diffX > 0) {  
                        onSwipeRight();  
                    } 否则 {  
                        onSwipeLeft();  
                    }  
                }  
            } 否则如果 (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) >  
                SWIPE_VELOCITY_THRESHOLD) {  
                if (diffY > 0) {  
                    onSwipeBottom();  
                } 否则 {  
                    onSwipeTop();  
                }  
            }  
            return true;  
        }  
  
        public void onSwipeRight() {}  
  
        public void onSwipeLeft() {}  
  
        public void onSwipeTop() {}  
    }  
}
```

```

public void onSwipeBottom() {
}

```

应用于视图...

```

view.setOnTouchListener(new OnSwipeListener(context) {
    public void onSwipeTop() {
        Log.d("OnSwipeListener", "onSwipeTop");
    }
    public void onSwipeRight() {
        Log.d("OnSwipeListener", "onSwipeRight");
    }
    public void onSwipeLeft() {
        Log.d("OnSwipeListener", "onSwipeLeft");
    }
    public void onSwipeBottom() {
        Log.d("OnSwipeListener", "onSwipeBottom");
    }
});

```

第138.2节：基本手势检测

```

public class GestureActivity extends Activity implements
    GestureDetector.OnDoubleTapListener,
    GestureDetector.OnGestureListener {

    private GestureDetector mGestureDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mGestureDetector = new GestureDetector(this, this);
        mGestureDetector.setOnDoubleTapListener(this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        mGestureDetector.onTouchEvent(event);
        return super.onTouchEvent(event);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        Log.d("GestureDetector", "onDown");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float velocityX, float
velocityY) {
        Log.d("GestureDetector", "onFling");
        return true;
    }

    @Override
    public void onLongPress(MotionEvent event) {

```

```

public void onSwipeBottom() {
}

```

Applied to a view...

```

view.setOnTouchListener(new OnSwipeListener(context) {
    public void onSwipeTop() {
        Log.d("OnSwipeListener", "onSwipeTop");
    }
    public void onSwipeRight() {
        Log.d("OnSwipeListener", "onSwipeRight");
    }
    public void onSwipeLeft() {
        Log.d("OnSwipeListener", "onSwipeLeft");
    }
    public void onSwipeBottom() {
        Log.d("OnSwipeListener", "onSwipeBottom");
    }
});

```

Section 138.2: Basic Gesture Detection

```

public class GestureActivity extends Activity implements
    GestureDetector.OnDoubleTapListener,
    GestureDetector.OnGestureListener {

    private GestureDetector mGestureDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mGestureDetector = new GestureDetector(this, this);
        mGestureDetector.setOnDoubleTapListener(this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        mGestureDetector.onTouchEvent(event);
        return super.onTouchEvent(event);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        Log.d("GestureDetector", "onDown");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float velocityX, float
velocityY) {
        Log.d("GestureDetector", "onFling");
        return true;
    }

    @Override
    public void onLongPress(MotionEvent event) {

```

```
Log.d("GestureDetector", "onLongPress");
}

@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
    Log.d("GestureDetector", "onScroll");
    return true;
}

@Override
public void onShowPress(MotionEvent event) {
    Log.d("GestureDetector", "onShowPress");
}

@Override
public boolean onSingleTapUp(MotionEvent event) {
    Log.d("GestureDetector", "onSingleTapUp");
    return true;
}

@Override
public boolean onDoubleTap(MotionEvent event) {
    Log.d("GestureDetector", "onDoubleTap");
    return true;
}

@Override
public boolean onDoubleTapEvent(MotionEvent event) {
    Log.d("GestureDetector", "onDoubleTapEvent");
    return true;
}

@Override
public boolean onSingleTapConfirmed(MotionEvent event) {
    Log.d("GestureDetector", "onSingleTapConfirmed");
    return true;
}
```

```
Log.d("GestureDetector", "onLongPress");
}

@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
    Log.d("GestureDetector", "onScroll");
    return true;
}

@Override
public void onShowPress(MotionEvent event) {
    Log.d("GestureDetector", "onShowPress");
}

@Override
public boolean onSingleTapUp(MotionEvent event) {
    Log.d("GestureDetector", "onSingleTapUp");
    return true;
}

@Override
public boolean onDoubleTap(MotionEvent event) {
    Log.d("GestureDetector", "onDoubleTap");
    return true;
}

@Override
public boolean onDoubleTapEvent(MotionEvent event) {
    Log.d("GestureDetector", "onDoubleTapEvent");
    return true;
}

@Override
public boolean onSingleTapConfirmed(MotionEvent event) {
    Log.d("GestureDetector", "onSingleTapConfirmed");
    return true;
}
```

第139章：休眠模式

第139.1节：以编程方式将Android应用程序加入白名单

将应用程序加入白名单不会禁用休眠模式，但你可以通过使用网络和保持唤醒锁来实现这一点。

以编程方式将Android应用程序加入白名单可以按如下方式进行：

```
boolean isIgnoringBatteryOptimizations = pm.isIgnoringBatteryOptimizations(getPackageName());  
if(!isIgnoringBatteryOptimizations){  
    Intent intent = new Intent();  
    intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);  
    intent.setData(Uri.parse("package:" + getPackageName()));  
    startActivityForResult(intent, MY_IGNORE_OPTIMIZATION_REQUEST);  
}
```

上述启动活动的结果可以通过以下代码验证：

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    if (requestCode == MY_IGNORE_OPTIMIZATION_REQUEST) {  
        PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);  
        boolean isIgnoringBatteryOptimizations =  
            pm.isIgnoringBatteryOptimizations(getPackageName());  
        if(isIgnoringBatteryOptimizations){  
            // 忽略电池优化  
        }else{  
            // 未忽略电池优化  
        }  
    }  
}
```

第139.2节：将应用排除在休眠模式之外

1. 打开手机设置
2. 打开电池
3. 打开菜单并选择“电池优化”
4. 从下拉菜单中选择“所有应用”
5. 选择你想加入白名单的应用
6. 选择“不优化”

现在该应用将显示在未优化的应用列表中。

应用可以通过调用isIgnoringBatteryOptimizations()来检查是否被加入白名单

Chapter 139: Doze Mode

Section 139.1: Whitelisting an Android application programmatically

Whitelisting won't disable the doze mode for your app, but you can do that by using network and hold-wake locks.

Whitelisting an Android application programmatically can be done as follows:

```
boolean isIgnoringBatteryOptimizations = pm.isIgnoringBatteryOptimizations(getPackageName());  
if(!isIgnoringBatteryOptimizations){  
    Intent intent = new Intent();  
    intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);  
    intent.setData(Uri.parse("package:" + getPackageName()));  
    startActivityForResult(intent, MY_IGNORE_OPTIMIZATION_REQUEST);  
}
```

The result of starting the activity above can be verified by the following code:

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    if (requestCode == MY_IGNORE_OPTIMIZATION_REQUEST) {  
        PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);  
        boolean isIgnoringBatteryOptimizations =  
            pm.isIgnoringBatteryOptimizations(getPackageName());  
        if(isIgnoringBatteryOptimizations){  
            // Ignoring battery optimization  
        }else{  
            // Not ignoring battery optimization  
        }  
    }  
}
```

Section 139.2: Exclude app from using doze mode

1. Open phone's settings
2. open battery
3. open menu and select "battery optimization"
4. from the dropdown menu select "all apps"
5. select the app you want to whitelist
6. select "don't optimize"

Now this app will show under not optimized apps.

An app can check whether it's whitelisted by calling isIgnoringBatteryOptimizations()

第140章：颜色

第140.1节：颜色操作

要操作颜色，我们将修改颜色的argb（透明度、红色、绿色和蓝色）值。

首先从您的颜色中提取RGB值。

```
int yourColor = Color.parseColor("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

现在你可以减少或增加红色、绿色和蓝色的数值，然后将它们组合成一个新的颜色：

```
int newColor = Color.rgb(red, green, blue);
```

或者如果你想添加一些透明度，可以在创建颜色时添加：

```
int newColor = Color.argb(alpha, red, green, blue);
```

Alpha 和 RGB 的数值范围应在 [0-225] 之间。

Chapter 140: Colors

Section 140.1: Color Manipulation

To manipulate colors we will modify the argb (Alpha, Red, Green and Blue) values of a color.

First extract RGB values from your color.

```
int yourColor = Color.parseColor("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Now you can reduce or increase red, green, and blue values and combine them to be a color again:

```
int newColor = Color.rgb(red, green, blue);
```

Or if you want to add some alpha to it, you can add it while creating the color:

```
int newColor = Color.argb(alpha, red, green, blue);
```

Alpha and RGB values should be in the range [0-225].

第141章：键盘

第141.1节：注册键盘打开和关闭的回调

思路是在每次变化前后测量布局，如果有显著变化，你可以基本确定是软键盘。

```
// 用于保存上一次内容布局高度的变量  
private int mLastContentHeight = 0;  
  
private ViewTreeObserver.OnGlobalLayoutListener keyboardLayoutListener = new  
ViewTreeObserver.OnGlobalLayoutListener() {  
    @Override public void onGlobalLayout() {  
        int currentContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();  
  
        if (mLastContentHeight > currentContentHeight + 100) {  
            Timber.d("onGlobalLayout: 键盘已打开");  
        } else if (currentContentHeight > mLastContentHeight + 100) {  
            Timber.d("onGlobalLayout: 键盘已关闭");  
            mLastContentHeight = currentContentHeight;  
        }  
    }  
};
```

然后在我们的 onCreate 中设置 mLastContentHeight 的初始值

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

并添加监听器

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

别忘了在销毁时移除监听器

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

第141.2节：当用户点击屏幕其他地方时隐藏键盘

在你的Activity中添加代码。

这对Fragment也适用，无需在Fragment中添加此代码。

```
@Override  
public boolean dispatchTouchEvent(MotionEvent ev) {  
    View view = getCurrentFocus();  
    if (view != null && (ev.getAction() == MotionEvent.ACTION_UP || ev.getAction() ==  
    MotionEvent.ACTION_MOVE) && view instanceof EditText &&  
    !view.getClass().getName().startsWith("android.webkit.")) {  
        int scrcoords[] = new int[2];  
        view.getLocationOnScreen(scrcoords);  
        float x = ev.getRawX() + view.getLeft() - scrcoords[0];  
        float y = ev.getRawY() + view.getTop() - scrcoords[1];  
        if (x < view.getLeft() || x > view.getRight() || y < view.getTop() || y > view.getBottom())
```

Chapter 141: Keyboard

Section 141.1: Register a callback for keyboard open and close

The idea is to measure a layout before and after each change and if there is a significant change you can be somewhat certain that its the softkeyboard.

```
// A variable to hold the last content layout height  
private int mLastContentHeight = 0;  
  
private ViewTreeObserver.OnGlobalLayoutListener keyboardLayoutListener = new  
ViewTreeObserver.OnGlobalLayoutListener() {  
    @Override public void onGlobalLayout() {  
        int currentContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();  
  
        if (mLastContentHeight > currentContentHeight + 100) {  
            Timber.d("onGlobalLayout: Keyboard is open");  
            mLastContentHeight = currentContentHeight;  
        } else if (currentContentHeight > mLastContentHeight + 100) {  
            Timber.d("onGlobalLayout: Keyboard is closed");  
            mLastContentHeight = currentContentHeight;  
        }  
    }  
};
```

then in our onCreate set the initial value for mLastContentHeight

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

and add the listener

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

don't forget to remove the listener on destroy

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Section 141.2: Hide keyboard when user taps anywhere else on the screen

Add code in your **Activity**.

This would work for **Fragment** also, **no need** to add this code in **Fragment**.

```
@Override  
public boolean dispatchTouchEvent(MotionEvent ev) {  
    View view = getCurrentFocus();  
    if (view != null && (ev.getAction() == MotionEvent.ACTION_UP || ev.getAction() ==  
    MotionEvent.ACTION_MOVE) && view instanceof EditText &&  
    !view.getClass().getName().startsWith("android.webkit.")) {  
        int scrcoords[] = new int[2];  
        view.getLocationOnScreen(scrcoords);  
        float x = ev.getRawX() + view.getLeft() - scrcoords[0];  
        float y = ev.getRawY() + view.getTop() - scrcoords[1];  
        if (x < view.getLeft() || x > view.getRight() || y < view.getTop() || y > view.getBottom())
```

```
((InputMethodManager)this.getSystemService(Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow(  
this.getWindow().getDecorView().getApplicationWindowToken()), 0);  
}  
return super.dispatchTouchEvent(ev);  
}
```

```
((InputMethodManager)this.getSystemService(Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow(  
this.getWindow().getDecorView().getApplicationWindowToken()), 0);  
}  
return super.dispatchTouchEvent(ev);  
}
```

第142章：RenderScript

RenderScript是一种脚本语言，允许您编写高性能的图形渲染和原始计算代码。它提供了一种编写性能关键代码的方法，系统随后会将其编译为可在处理器上运行的本地代码。这个处理器可以是CPU、多核CPU，甚至是GPU。最终运行在哪个处理器上取决于许多开发者无法轻易获得的因素，同时也取决于内部平台编译器支持的架构。

第142.1节：入门

RenderScript是一个框架，允许在Android上进行高性能的并行计算。您编写的脚本将会在所有可用的处理器（例如CPU、GPU等）上并行执行，使您能够专注于想要完成的任务，而无需关心任务如何被调度和执行。

脚本是用基于C99的语言编写的（C99是C编程语言标准的一个旧版本）。

对于每个脚本，都会创建一个Java类，允许你在Java代码中轻松地与RenderScript进行交互。

设置你的项目

在你的应用中访问RenderScript有两种不同的方式，使用Android框架库或支持库。即使你不打算支持API级别11之前的设备，也应始终使用支持库实现，因为它确保了在许多不同设备上的兼容性。要使用支持库实现，你需要使用至少版本为18.1.0的构建工具！

现在让我们设置应用的build.gradle文件：

```
android {  
    compileSdkVersion 24  
    buildToolsVersion '24.0.1'  
  
    defaultConfig {  
        minSdkVersion 8  
        targetSdkVersion 24  
  
        renderscriptTargetApi 18  
        renderscriptSupportModeEnabled true  
    }  
}
```

- `renderscriptTargetApi`：应设置为提供你所需所有RenderScript功能的最低API版本。
- `renderscriptSupportModeEnabled`：启用支持库RenderScript实现的使用。

RenderScript的工作原理

一个典型的RenderScript由两部分组成：内核（Kernels）和函数（Functions）。函数顾名思义——接受输入，对输入进行处理并返回输出。内核则是RenderScript真正强大之处所在。

内核（Kernel）是一个函数，它会针对Allocation中的每个元素执行。一个Allocation可以用来传递数据，比如Bitmap或byte数组给RenderScript，同时也用于从内核获取结果。

内核可以接受一个Allocation作为输入，另一个作为输出，或者只修改一个Allocation中的数据。

Chapter 142: RenderScript

RenderScript is a scripting language that allows you to write high performance graphic rendering and raw computational code. It provides a means of writing performance critical code that the system later compiles to native code for the processor it can run on. This could be the CPU, a multi-core CPU, or even the GPU. Which it ultimately runs on depends on many factors that aren't readily available to the developer, but also depends on what architecture the internal platform compiler supports.

Section 142.1: Getting Started

RenderScript is a framework to allow high performance parallel computation on Android. Scripts you write will be executed across all available processors (e.g. CPU, GPU etc) in parallel allowing you to focus on the task you want to achieve instead of how it is scheduled and executed.

Scripts are written in a C99 based language (C99 being an old version of the C programming language standard). For each Script a Java class is created which allows you to easily interact with RenderScript in your Java code.

Setting up your project

There exist two different ways to access RenderScript in your app, with the Android Framework libraries or the Support Library. Even if you don't want to target devices before API level 11 you should always use the Support Library implementation because it ensures devices compatibility across many different devices. To use the support library implementation you need to use at least build tools version 18.1.0!

Now lets setup the build.gradle file of your application:

```
android {  
    compileSdkVersion 24  
    buildToolsVersion '24.0.1'  
  
    defaultConfig {  
        minSdkVersion 8  
        targetSdkVersion 24  
  
        renderscriptTargetApi 18  
        renderscriptSupportModeEnabled true  
    }  
}
```

- `renderscriptTargetApi`: This should be set to the version earliest API level which provides all RenderScript functionality you require.
- `renderscriptSupportModeEnabled`: This enables the use of the Support Library RenderScript implementation.

How RenderScript works

A typical RenderScript consists of two things: Kernels and Functions. A function is just what it sounds like - it accepts an input, does something with that input and returns an output. A Kernel is where the real power of RenderScript comes from.

A Kernel is a function which is executed against every element inside an Allocation. An Allocation can be used to pass data like a Bitmap or a byte array to a RenderScript and they are also used to get a result from a Kernel. Kernels can either take one Allocation as input and another as output or they can modify the data inside just one Allocation.

你可以编写自己的内核，但也有许多预定义的内核可供使用，以执行常见操作，比如高斯图像模糊。

如前所述，每个RenderScript文件都会生成一个用于交互的类。这些类总是以ScriptC_为前缀，后跟RenderScript文件的名称。例如，如果你的RenderScript文件名为example，那么生成的Java类将被命名为ScriptC_example。所有预定义的脚本都以Script为前缀——例如，高斯图像模糊脚本名为ScriptIntrinsicBlur。

编写你的第一个RenderScript

以下示例基于GitHub上的一个示例。它通过修改图像的饱和度来执行基本的图像处理。你可以在[here](#)找到源代码，如果你想自己尝试，可以查看。下面是结果的一个快速动图：



RenderScript模板代码

RenderScript文件位于项目中的文件夹src/main/rs下。每个文件的扩展名为.rs，且顶部必须包含两个#pragma语句：

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- #pragma version(1)：这可以用来设置你使用的 RenderScript 版本。目前只有版本 1。
- #pragma rs java_package_name(your.package.name)：这可以用来设置生成的 Java 类的包名，该类用于与此特定 RenderScript 交互。

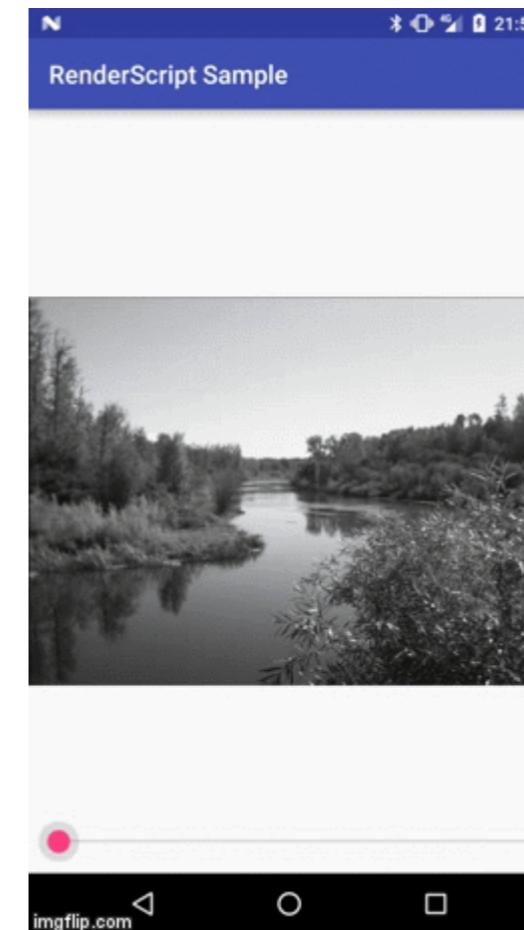
你通常还应该在每个 RenderScript 文件中设置另一个 #pragma，它用于设置浮点精度。

You can write your own Kernels, but there are also many predefined Kernels which you can use to perform common operations like a Gaussian Image Blur.

As already mentioned for every RenderScript file a class is generated to interact with it. These classes always start with the prefix ScriptC_ followed by the name of the RenderScript file. For example if your RenderScript file is called example then the generated Java class will be called ScriptC_example. All predefined Scripts just start with the prefix Script - for example the Gaussian Image Blur Script is called ScriptIntrinsicBlur.

Writing your first RenderScript

The following example is based on an example on GitHub. It performs basic image manipulation by modifying the saturation of an image. You can find the source code [here](#) and check it out if you want to play around with it yourself. Here's a quick gif of what the result is supposed to look like:



RenderScript Boilerplate

RenderScript files reside in the folder src/main/rs in your project. Each file has the file extension .rs and has to contain two #pragma statements at the top:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- #pragma version(1)：This can be used to set the version of RenderScript you are using. Currently there is only version 1.
- #pragma rs java_package_name(your.package.name)：This can be used to set the package name of the Java class generated to interact with this particular RenderScript.

There is another #pragma you should usually set in each of your RenderScript files and it is used to set the floating

你可以将浮点精度设置为三个不同的级别：

- `#pragma rs_fp_full`：这是最严格的设置，具有最高精度，如果不指定任何内容，默认值即为此。若需要高浮点精度，应使用此设置。
- `#pragma rs_fp_relaxed`：这保证了不那么高的浮点精度，但在某些架构上启用许多优化，可能使脚本运行更快。
- `#pragma rs_fp_imprecise`：这保证了更低的精度，适用于浮点精度对脚本不重要的情况。

大多数脚本可以使用 `#pragma rs_fp_relaxed`，除非你确实需要高浮点精度。

全局变量

现在，就像在C代码中一样，你可以定义全局变量或常量：

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

变量gMonoMult的类型是float3。这意味着它是一个由3个浮点数组成的向量。另一个float变量名为saturationValue不是常量，因此你可以在运行时将其设置为你想要的值。你可以在你的内核（Kernel）或函数中使用这样的变量，因此它们是向你的RenderScripts传入输入或接收输出的另一种方式。对于每个非常量变量，相关的Java类中将生成对应的getter和setter方法。

内核（Kernels）

现在让我们开始实现内核。为了本例的目的，我不会解释用于修改图像饱和度的数学原理，而是重点讲解如何实现内核以及如何使用它。在本章末尾，我会简要说明这个内核中的代码实际在做什么。

内核概述

我们先来看一下源代码：

```
uchar4 __attribute__((kernel)) saturation(uchar4 in) {
    float4 f4 = rsUnpackColor8888(in);
    float3 dotVector = dot(f4.rgb, gMonoMult);
    float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
    return rsPackColorTo8888(newColor);
}
```

如你所见，它看起来像一个普通的C函数，唯一的例外是：返回类型和方法名之间的`__attribute__((kernel))`。这告诉RenderScript该方法是一个内核（Kernel）。你可能还会注意到该方法接受一个uchar4参数并返回另一个uchar4值。就像我们在前一章讨论的float3变量一样，uchar4是一个向量。它包含4个uchar值，这些值是范围在0到255之间的字节值。

你可以通过多种方式访问这些单独的值，例如in.r会返回对应像素红色通道的字节。我们使用uchar4是因为每个像素由4个值组成——r代表红色，g代表绿色，b代表蓝色，a代表透明度——你可以用这种简写方式访问它们。RenderScript还允许你从一个向量中取任意数量的值并创建另一个向量。例如in.rgb会返回一个uchar3值，它只包含像素的红、绿、蓝部分，不包括透明度值。

在运行时，RenderScript会为图像的每个像素调用这个内核方法，这就是返回值和

point precision. You can set the floating point precision to three different levels:

- `#pragma rs_fp_full`: This is the strictest setting with the highest precision and it is also the default value if you don't specify anything. You should use this if you require high floating point precision.
- `#pragma rs_fp_relaxed`: This ensures not quite as high floating point precision, but on some architectures it enables a bunch of optimizations which can cause your scripts to run faster.
- `#pragma rs_fp_imprecise`: This ensures even less precision and should be used if floating point precision does not really matter to your script.

Most scripts can just use `#pragma rs_fp_relaxed` unless you really need high floating point precision.

Global Variables

Now just like in C code you can define global variables or constants:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

The variable gMonoMult is of type float3. This means it is a vector consisting of 3 float numbers. The other `float` variable called saturationValue is not constant, therefore you can set it at runtime to a value you like. You can use variables like this in your Kernels or functions and therefore they are another way to give input to or receive output from your RenderScripts. For each not constant variable a getter and setter method will be generated on the associated Java class.

Kernels

But now lets get started implementing the Kernel. For the purposes of this example I am not going to explain the math used in the Kernel to modify the saturation of the image, but instead will focus on how to implement a Kernel and how to use it. At the end of this chapter I will quickly explain what the code in this Kernel is actually doing.

Kernels in general

Let's take a look at the source code first:

```
uchar4 __attribute__((kernel)) saturation(uchar4 in) {
    float4 f4 = rsUnpackColor8888(in);
    float3 dotVector = dot(f4.rgb, gMonoMult);
    float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
    return rsPackColorTo8888(newColor);
}
```

As you can see it looks like a normal C function with one exception: The `__attribute__((kernel))` between the return type and method name. This is what tells RenderScript that this method is a Kernel. Another thing you might notice is that this method accepts a uchar4 parameter and returns another uchar4 value. uchar4 is - like the float3 variable we discussed in the chapter before - a vector. It contains 4 uchar values which are just byte values in the range from 0 to 255.

You can access these individual values in many different ways, for example `in.r` would return the byte which corresponds to the red channel of a pixel. We use a uchar4 since each pixel is made up of 4 values - r for red, g for green, b for blue and a for alpha - and you can access them with this shorthand. RenderScript also allows you to take any number of values from a vector and create another vector with them. For example `in.rgb` would return a uchar3 value which just contains the red, green and blue parts of the pixel without the alpha value.

At runtime RenderScript will call this Kernel method for each pixel of an image which is why the return value and

参数仅是一个uchar4值的原因。RenderScript会在所有可用处理器上并行运行许多这样的调用，这也是RenderScript如此强大的原因。这也意味着你不必担心线程或线程安全问题，你只需实现对每个像素的操作，RenderScript会处理剩下的部分。

在Java中调用内核时，你需要提供两个Allocation变量，一个包含输入数据，另一个用于接收输出。你的内核方法会针对输入Allocation中的每个值被调用，并将结果写入输出Allocation。

RenderScript运行时API方法

在上述内核中使用了一些开箱即用的方法。RenderScript提供了许多此类方法，它们对于你使用RenderScript几乎所有操作都至关重要。其中包括执行数学运算的方法，如sin()，以及辅助方法，如mix()，它根据另一个值混合两个值。

但也有处理向量、四元数和矩阵时更复杂操作的方法。

官方的RenderScript Runtime API Reference是了解特定方法或查找执行常见操作（如计算矩阵点积）的方法的最佳资源。你可以在[here](#)找到这份文档。

内核实现

现在让我们来看一下这个内核具体在做什么。以下是内核中的第一行代码：

```
float4 f4 = rsUnpackColor8888(in);
```

第一行调用了内置方法 `rsUnpackColor8888()`，它将 uchar4 值转换为 float4 值。每个颜色通道也被转换到范围 0.0f - 1.0f，其中 0.0f 对应字节值 0，1.0f 对应 255。这样做的主要目的是让内核中的所有数学运算变得更容易。

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

下一行使用内置方法 `dot()` 来计算两个向量的点积。gMonoMult 是我们在前面几章定义的常量值。由于计算点积时两个向量必须长度相同，且我们只想影响像素的颜色通道而非透明度通道，因此使用简写 `.rgb` 来获取一个新的 float3 向量，该向量仅包含红、绿、蓝三个颜色通道。还记得学校里点积的计算方法的朋友会很快注意到，点积应该返回一个值而不是向量。但在上面的代码中，我们将结果赋值给了一个 float3 向量。这又是 RenderScript 的一个特性。当你将一个一维数值赋给向量时，向量中的所有元素都会被设置为该值。例如，下面的代码片段会将 2.0f 赋值给 float3 向量中的三个值：

```
float3 example = 2.0f;
```

因此，上面点积的结果会被赋值给 float3 向量中的每个元素。

现在到了我们实际使用全局变量 `saturationLevel` 来修改图像饱和度的部分：

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

这使用了内置方法 `mix()`，将原始颜色与上面创建的点积向量混合在一起。它们如何混合由全局变量 `saturationLevel` 决定。因此，`saturationLevel` 为 0.0f 时，结果颜色将不包含原始颜色值，只包含 `dotVector` 中的值，这会导致图像变成黑白或灰度图像。值为 1.0f 时，结果颜色将

parameter are just one uchar4 value. RenderScript will run many of these calls in parallel on all available processors which is why RenderScript is so powerful. This also means that you don't have to worry about threading or thread safety, you can just implement whatever you want to do to each pixel and RenderScript takes care of the rest.

When calling a Kernel in Java you supply two Allocation variables, one which contains the input data and another one which will receive the output. Your Kernel method will be called for each value in the input Allocation and will write the result to the output Allocation.

RenderScript Runtime API methods

In the Kernel above a few methods are used which are provided out of the box. RenderScript provides many such methods and they are vital for almost anything you are going to do with RenderScript. Among them are methods to do math operations like `sin()` and helper methods like `mix()` which mixes two values according to another values. But there are also methods for more complex operations when dealing with vectors, quaternions and matrices.

The official [RenderScript Runtime API Reference](#) is the best resource out there if you want to know more about a particular method or are looking for a specific method which performs a common operation like calculating the dot product of a matrix. You can find this documentation [here](#).

Kernel Implementation

Now let's take a look at the specifics of what this Kernel is doing. Here's the first line in the Kernel:

```
float4 f4 = rsUnpackColor8888(in);
```

The first line calls the built in method `rsUnpackColor8888()` which transforms the uchar4 value to a float4 values. Each color channel is also transformed to the range 0.0f - 1.0f where 0.0f corresponds to a byte value of 0 and 1.0f to 255. The main purpose of this is to make all the math in this Kernel a lot simpler.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

This next line uses the built in method `dot()` to calculate the dot product of two vectors. gMonoMult is a constant value we defined a few chapters above. Since both vectors need to be of the same length to calculate the dot product and also since we just want to affect the color channels and not the alpha channel of a pixel we use the shorthand `.rgb` to get a new float3 vector which just contains the red, green and blue color channels. Those of us who still remember from school how the dot product works will quickly notice that the dot product should return just one value and not a vector. Yet in the code above we are assigning the result to a float3 vector. This is again a feature of RenderScript. When you assign a one dimensional number to a vector all elements in the vector will be set to this value. For example the following snippet will assign 2.0f to each of the three values in the float3 vector:

```
float3 example = 2.0f;
```

So the result of the dot product above is assigned to each element in the float3 vector above.

Now comes the part in which we actually use the global variable `saturationLevel` to modify the saturation of the image:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

This uses the built in method `mix()` to mix together the original color with the dot product vector we created above. How they are mixed together is determined by the global `saturationLevel` variable. So a `saturationLevel` of 0.0f will cause the resulting color to have no part of the original color values and will only consist of values in the `dotVector` which results in a black and white or grayed out image. A value of 1.0f will cause the resulting color to

完全由原始颜色值组成，超过1.0f的值会使原始颜色被放大，变得更亮更强烈。

```
return rsPackColorTo8888(newColor);
```

这是内核的最后部分。rsPackColorTo8888()将float3向量转换回uchar4值，然后返回。结果的字节值被限制在0到255之间，因此大于1.0f的浮点值将对应字节值255，小于0.0的值将对应字节值0。

这就是整个内核的实现。现在只剩下一部分：如何在Java中调用内核。

在Java中调用RenderScript

基础知识

如上所述，每个RenderScript文件都会生成一个Java类，允许你与脚本交互。这些文件以ScriptC_为前缀，后跟RenderScript文件名。要创建这些类的实例，首先需要一个RenderScript类的实例：

```
final RenderScript renderScript = RenderScript.create(context);
```

静态方法create()可以用来从Context创建一个RenderScript实例。然后你可以实例化为你的脚本生成的Java类。如果你将RenderScript文件命名为saturation.rs，那么类名将是ScriptC_saturation：

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

在这个类上，你现在可以设置饱和度级别并调用内核。为变量saturationLevel生成的设置方法将以set_为前缀，后跟变量名：

```
script.set_saturationLevel(1.0f);
```

还有一个以get_为前缀的获取方法，允许你获取当前设置的饱和度级别：

```
float saturationLevel = script.get_saturationLevel();
```

你在RenderScript中定义的内核以forEach_为前缀，后跟内核方法名。

我们编写的内核期望输入参数为一个Allocation，输出参数也是一个Allocation：

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

输入Allocation需要包含输入图像，在forEach_saturation方法完成后，输出allocation将包含修改后的图像数据。

一旦你有了一个Allocation实例，就可以使用方法

copyFrom()和copyTo()在这些Allocations之间复制数据。例如，你可以通过调用以下方法将新图像复制到你的输入Allocation中：

```
inputAllocation.copyFrom(inputBitmap);
```

同样，你可以通过在输出Allocation上调用copyTo()来获取结果图像：

```
outputAllocation.copyTo(outputBitmap);
```

创建 Allocation 实例

be completely made up of the original color values and values above 1.0f will multiply the original colors to make them more bright and intense.

```
return rsPackColorTo8888(newColor);
```

This is the last part in the Kernel. [rsPackColorTo8888\(\)](#) transforms the float3 vector back to a uchar4 value which is then returned. The resulting byte values are clamped to a range between 0 and 255, so float values higher than 1.0f will result in a byte value of 255 and values lower than 0.0 will result in a byte value of 0.

And that is the whole Kernel implementation. Now there is only one part remaining: How to call a Kernel in Java.

Calling RenderScript in Java Basics

As was already mentioned above for each RenderScript file a Java class is generated which allows you to interact with your scripts. These files have the prefix ScriptC_ followed by the name of the RenderScript file. To create an instance of these classes you first need an instance of the RenderScript class:

```
final RenderScript renderScript = RenderScript.create(context);
```

The static method `create()` can be used to create a RenderScript instance from a [Context](#). You can then instantiate the Java class which was generated for your script. If you called the RenderScript file `saturation.rs` then the class will be called `ScriptC_saturation`:

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

On this class you can now set the saturation level and call the Kernel. The setter which was generated for the `saturationLevel` variable will have the prefix `set_` followed by the name of the variable:

```
script.set_saturationLevel(1.0f);
```

There is also a getter prefixed with `get_` which allows you to get the saturation level currently set:

```
float saturationLevel = script.get_saturationLevel();
```

Kernels you define in your RenderScript are prefixed with `forEach_` followed by the name of the Kernel method. The Kernel we have written expects an input Allocation and an output Allocation as its parameters:

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

The input Allocation needs to contain the input image, and after the `forEach_saturation` method has finished the output allocation will contain the modified image data.

Once you have an Allocation instance you can copy data from and to those Allocations by using the methods `copyFrom()` and `copyTo()`. For example you can copy a new image into your input `Allocation by calling:

```
inputAllocation.copyFrom(inputBitmap);
```

The same way you can retrieve the result image by calling `copyTo()` on the output Allocation:

```
outputAllocation.copyTo(outputBitmap);
```

Creating Allocation instances

创建Allocation有多种方式。一旦你有了一个Allocation实例，就可以像上面解释的那样使用copyTo()和copyFrom()在这些Allocations之间复制新数据，但要最初创建它们，你必须确切知道你正在处理哪种类型的数据。让我们从输入Allocation开始：

我们可以使用静态方法createFromBitmap()快速从一个Bitmap创建输入Allocation：

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

在这个例子中，输入图像从未改变，因此我们不需要再次修改输入Allocation。每当饱和度等级(saturationLevel)变化时，我们都可重复使用它来创建新的输出Bitmap。

创建输出Allocation稍微复杂一些。首先我们需要创建一个称为Type的东西。Type用于告诉Allocation它处理的数据类型。通常使用Type.Builder类来快速创建合适的Type。先来看一下代码：

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
    .setX(inputBitmap.getWidth())
    .setY(inputBitmap.getHeight())
    .create();
```

我们使用的是普通的32位（换句话说每像素4字节）Bitmap，具有4个颜色通道。这就是为什么我们选择Element.RGBA_8888来创建Type。然后我们使用方法setX()和setY()来设置输出图像的宽度和高度，使其与输入图像大小相同。方法create()则根据我们指定的参数创建了Type。

一旦我们有了正确的Type，就可以使用静态方法createTyped()来创建输出Allocation：

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

现在我们几乎完成了。我们还需要一个输出Bitmap，用于从输出Allocation复制数据。为此，我们使用静态方法createBitmap()来创建一个新的空白Bitmap，其大小和配置与输入Bitmap相同。

```
final Bitmap outputBitmap = Bitmap.createBitmap(
    inputBitmap.getWidth(),
    inputBitmap.getHeight(),
    inputBitmap.getConfig()
);
```

至此，我们已经拥有执行 RenderScript 所需的所有拼图块。

完整示例

现在让我们将所有内容整合到一个示例中：

```
// 创建 RenderScript 实例
final RenderScript renderScript = RenderScript.create(context);

// 创建输入 Allocation
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, inputBitmap);

// 创建输出 Type。
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
    .setX(inputBitmap.getWidth())
    .setY(inputBitmap.getHeight())
    .create();
```

There are many ways to create an Allocation. Once you have an Allocation instance you can copy new data from and to those Allocations with copyTo() and copyFrom() like explained above, but to create them initially you have to know with what kind of data you are exactly working with. Let's start with the input Allocation:

We can use the static method createFromBitmap() to quickly create our input Allocation from a Bitmap:

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

In this example the input image never changes so we never need to modify the input Allocation again. We can reuse it each time the saturationLevel changes to create a new output Bitmap.

Creating the output Allocation is a little more complex. First we need to create what's called a Type. A Type is used to tell an Allocation with what kind of data it's dealing with. Usually one uses the Type.Builder class to quickly create an appropriate Type. Let's take a look at the code first:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
    .setX(inputBitmap.getWidth())
    .setY(inputBitmap.getHeight())
    .create();
```

We are working with a normal 32 bit (or in other words 4 byte) per pixel Bitmap with 4 color channels. That's why we are choosing Element.RGBA_8888 to create the Type. Then we use the methods setX() and setY() to set the width and height of the output image to the same size as the input image. The method create() then creates the Type with the parameters we specified.

Once we have the correct Type we can create the output Allocation with the static method createTyped():

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Now we are almost done. We also need an output Bitmap in which we can copy the data from the output Allocation. To do this we use the static method createBitmap() to create a new empty Bitmap with the same size and configuration as the input Bitmap.

```
final Bitmap outputBitmap = Bitmap.createBitmap(
    inputBitmap.getWidth(),
    inputBitmap.getHeight(),
    inputBitmap.getConfig()
);
```

And with that we have all the puzzle pieces to execute our RenderScript.

Full example

Now let's put all this together in one example:

```
// Create the RenderScript instance
final RenderScript renderScript = RenderScript.create(context);

// Create the input Allocation
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, inputBitmap);

// Create the output Type.
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
    .setX(inputBitmap.getWidth())
    .setY(inputBitmap.getHeight())
    .create();
```

```

// 并使用该 Type 创建输出 Allocation
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);

// 从输入 Bitmap 创建一个空的输出 Bitmap
final Bitmap outputBitmap = Bitmap.createBitmap(
    inputBitmap.getWidth(),
    inputBitmap.getHeight(),
    inputBitmap.getConfig()
);

// 创建我们的脚本实例
final ScriptC_saturation 脚本 = new ScriptC_saturation(renderScript);

// 设置饱和度等级
script.set_saturationLevel(2.0f);

// 执行内核
script.forEach_saturation(inputAllocation, outputAllocation);

// 将结果数据复制到输出位图
outputAllocation.copyTo(outputBitmap);

// 在某处显示结果位图
someImageView.setImageBitmap(outputBitmap);

```

结论

通过本介绍，你应该已经准备好编写自己的 RenderScript 内核来进行简单的图像处理。不过，有几点需要注意：

- **RenderScript 仅适用于应用程序项目:** 目前 RenderScript 文件不能作为库项目的一部分。
- **注意内存 :** RenderScript 非常快，但也可能占用大量内存。任何时候都不应有超过一个RenderScript实例。你还应该尽可能重用。
通常你只需要创建一次你的Allocation实例，之后可以重复使用。输出的Bitmaps或脚本实例也是如此。尽可能重用。
- **在后台进行工作 :** RenderScript虽然非常快，但绝不是瞬时完成的。任何内核，尤其是复杂的内核，都应该在UI线程之外执行，比如在AsyncTask或类似机制中。然而，大多数情况下你不必担心内存泄漏。所有与RenderScript相关的类只使用应用程序的Context，因此不会导致内存泄漏。但你仍需注意常见的问题，比如泄漏你自己使用的View、Activity或任何Context实例！
- **使用内置功能 :** 有许多预定义的脚本可以执行图像模糊、混合、转换、调整大小等任务。还有许多内置方法可以帮助你实现内核。
很可能如果你想做某件事，已经有脚本或方法实现了你想做的功能。不要重复造轮子。

如果你想快速入门并尝试实际代码，我建议你看看示例GitHub项目，该项目实现了本教程中讨论的具体示例。你可以在[here](#)找到该项目。

祝你玩得开心，使用RenderScript！

第142.2节：模糊一个视图

BlurBitmapTask.java

```

public class BlurBitmapTask extends AsyncTask<Bitmap, Void, Bitmap> {
    private final WeakReference<ImageView> imageViewReference;
    private final RenderScript renderScript;

```

```

// And use the Type to create an output Allocation
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);

// Create an empty output Bitmap from the input Bitmap
final Bitmap outputBitmap = Bitmap.createBitmap(
    inputBitmap.getWidth(),
    inputBitmap.getHeight(),
    inputBitmap.getConfig()
);

// Create an instance of our script
final ScriptC_saturation script = new ScriptC_saturation(renderScript);

// Set the saturation level
script.set_saturationLevel(2.0f);

// Execute the Kernel
script.forEach_saturation(inputAllocation, outputAllocation);

// Copy the result data to the output Bitmap
outputAllocation.copyTo(outputBitmap);

// Display the result Bitmap somewhere
someImageView.setImageBitmap(outputBitmap);

```

Conclusion

With this introduction you should be all set to write your own RenderScript Kernels for simple image manipulation. However there are a few things you have to keep in mind:

- **RenderScript only works in Application projects:** Currently RenderScript files cannot be part of a library project.
- **Watch out for memory:** RenderScript is very fast, but it can also be memory intensive. There should never be more than one instance of RenderScript at any time. You should also reuse as much as possible.
Normally you just need to create your Allocation instances once and can reuse them in the future. The same goes for output Bitmaps or your script instances. Reuse as much as possible.
- **Do your work in the background:** Again RenderScript is very fast, but not instant in any way. Any Kernel, especially complex ones should be executed off the UI thread in an AsyncTask or something similar. However for the most part you don't have to worry about memory leaks. All RenderScript related classes only use the application Context and therefore don't cause memory leaks. But you still have to worry about the usual stuff like leaking View, Activity or any Context instance which you use yourself!
- **Use built in stuff:** There are many predefined scripts which perform tasks like image blurring, blending, converting, resizing. And there are many more built in methods which help you implement your kernels. Chances are that if you want to do something there is either a script or method which already does what you are trying to do. Don't reinvent the wheel.

If you want to quickly get started and play around with actual code I recommend you take a look at the example GitHub project which implements the exact example talked about in this tutorial. You can find the project [here](#). Have fun with RenderScript!

Section 142.2: Blur a View

BlurBitmapTask.java

```

public class BlurBitmapTask extends AsyncTask<Bitmap, Void, Bitmap> {
    private final WeakReference<ImageView> imageViewReference;
    private final RenderScript renderScript;

```

```

private boolean shouldRecycleSource = false;

public BlurBitmapTask(@NonNull Context context, @NonNull ImageView imageView) {
    // 使用 WeakReference 确保
    // ImageView 可以被垃圾回收
    imageViewReference = new WeakReference<>(imageView);
    renderScript = RenderScript.create(context);
}

// 在后台解码图像。
@Override
protected Bitmap doInBackground(Bitmap... params) {
    Bitmap bitmap = params[0];
    return blurBitmap(bitmap);
}

// 完成后，检查 ImageView 是否仍然存在并设置位图。
@Override
protected void onPostExecute(Bitmap bitmap) {
    if (bitmap == null || isCancelled()) {
        return;
    }

    final ImageView imageView = imageViewReference.get();
    if (imageView == null) {
        return;
    }

    imageView.setImageBitmap(bitmap);
}

public Bitmap 模糊位图(Bitmap 位图) {
    // https://plus.google.com/+MarioViviani/posts/fhuzYkji9zz

    //让我们创建一个与要模糊的位图大小相同的空位图
    Bitmap 输出位图 = Bitmap.createBitmap(位图.getWidth(), 位图.getHeight(),
        Bitmap.Config.ARGB_8888);

    //实例化一个新的 Renderscript

    //使用 Renderscript 创建一个内置模糊脚本
    ScriptIntrinsicBlur 模糊脚本 = ScriptIntrinsicBlur.create(renderScript,
        Element.U8_4(renderScript));

    //使用 Renderscript 和输入/输出位图创建输入/输出分配
    Allocation 输入分配 = Allocation.createFromBitmap(renderScript, 位图);
    Allocation 输出分配 = Allocation.createFromBitmap(renderScript, 输出位图);

    //设置模糊半径
    模糊脚本.setRadius(25.f);

    //执行Renderscript
    blurScript.setInput(allIn);
    blurScript.forEach(allOut);

    //将out Allocation创建的最终位图复制到outBitmap
    allOut.copyTo(outBitmap);

    //回收原始位图
    //不，我们也在使用原始位图：/
    if (shouldRecycleSource) {

```

```

private boolean shouldRecycleSource = false;

public BlurBitmapTask(@NonNull Context context, @NonNull ImageView imageView) {
    // Use a WeakReference to ensure
    // the ImageView can be garbage collected
    imageViewReference = new WeakReference<>(imageView);
    renderScript = RenderScript.create(context);
}

// Decode image in background.
@Override
protected Bitmap doInBackground(Bitmap... params) {
    Bitmap bitmap = params[0];
    return blurBitmap(bitmap);
}

// Once complete, see if ImageView is still around and set bitmap.
@Override
protected void onPostExecute(Bitmap bitmap) {
    if (bitmap == null || isCancelled()) {
        return;
    }

    final ImageView imageView = imageViewReference.get();
    if (imageView == null) {
        return;
    }

    imageView.setImageBitmap(bitmap);
}

public Bitmap blurBitmap(Bitmap bitmap) {
    // https://plus.google.com/+MarioViviani/posts/fhuzYkji9zz

    //Let's create an empty bitmap with the same size of the bitmap we want to blur
    Bitmap outBitmap = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(),
        Bitmap.Config.ARGB_8888);

    //Instantiate a new Renderscript

    //Create an Intrinsic Blur Script using the Renderscript
    ScriptIntrinsicBlur blurScript = ScriptIntrinsicBlur.create(renderScript,
        Element.U8_4(renderScript));

    //Create the in/out Allocations with the Renderscript and the in/out bitmaps
    Allocation allIn = Allocation.createFromBitmap(renderScript, bitmap);
    Allocation allOut = Allocation.createFromBitmap(renderScript, outBitmap);

    //Set the radius of the blur
    blurScript.setRadius(25.f);

    //Perform the Renderscript
    blurScript.setInput(allIn);
    blurScript.forEach(allOut);

    //Copy the final bitmap created by the out Allocation to the outBitmap
    allOut.copyTo(outBitmap);

    // recycle the original bitmap
    // nope, we are using the original bitmap as well :/
    if (shouldRecycleSource) {

```

```

bitmap.recycle();
}

//完成所有操作后，销毁RenderScript。
renderScript.destroy();

return outBitmap;
}

public boolean isShouldRecycleSource() {
    return shouldRecycleSource;
}

public void setShouldRecycleSource(boolean shouldRecycleSource) {
    this.shouldRecycleSource = shouldRecycleSource;
}
}

```

用法：

```

ImageView imageViewOverlayOnViewToBeBlurred
.setImageDrawable(ContextCompat.getDrawable(this, android.R.color.transparent));
View viewToBeBlurred.setDrawingCacheQuality(View.DRAWING_CACHE_QUALITY_LOW);
viewToBeBlurred.setDrawingCacheEnabled(true);
BlurBitmapTask blurBitmapTask = new BlurBitmapTask(this, imageViewOverlayOnViewToBeBlurred);
blurBitmapTask.execute(Bitmap.createBitmap(viewToBeBlurred.getDrawingCache()));
viewToBeBlurred.setDrawingCacheEnabled(false);

```

第142.3节：模糊图像

本示例演示如何使用RenderScript API对图像（使用Bitmap）进行模糊处理。本示例使用android RenderScript API (API >= 17) 提供的 [ScriptIntrinsicBlur](#)。

```

public class BlurProcessor {

    private RenderScript rs;
    private Allocation inAllocation;
    private Allocation outAllocation;
    private int width;
    private int height;

    private ScriptIntrinsicBlur blurScript;

    public BlurProcessor(RenderScript rs) {
        this.rs = rs;
    }

    public void 初始化(int 宽度, int 高度) {
        blurScript = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
        blurScript.setRadius(7f); // 设置模糊半径。最大值为25

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }

        // 位图必须具有 ARGB_8888 配置才能使用此类型
        Type bitmapType = new Type.Builder(rs, Element.RGBA_8888(rs))
            .setX(宽度)
            .setY(高度)
            .setMipmaps(false) // 我们使用的是 MipmapControl.MIPMAP_NONE
    }
}

```

```

bitmap.recycle();
}

//After finishing everything, we destroy the RenderScript.
renderScript.destroy();

return outBitmap;
}

public boolean isShouldRecycleSource() {
    return shouldRecycleSource;
}

public void setShouldRecycleSource(boolean shouldRecycleSource) {
    this.shouldRecycleSource = shouldRecycleSource;
}
}

```

Usage:

```

ImageView imageViewOverlayOnViewToBeBlurred
.setImageDrawable(ContextCompat.getDrawable(this, android.R.color.transparent));
View viewToBeBlurred.setDrawingCacheQuality(View.DRAWING_CACHE_QUALITY_LOW);
viewToBeBlurred.setDrawingCacheEnabled(true);
BlurBitmapTask blurBitmapTask = new BlurBitmapTask(this, imageViewOverlayOnViewToBeBlurred);
blurBitmapTask.execute(Bitmap.createBitmap(viewToBeBlurred.getDrawingCache()));
viewToBeBlurred.setDrawingCacheEnabled(false);

```

Section 142.3: Blur an image

This example demonstrates how to use RenderScript API to blur an image (using Bitmap). This example uses [ScriptIntrinsicBlur](#) provided by android RenderScript API (API >= 17).

```

public class BlurProcessor {

    private RenderScript rs;
    private Allocation inAllocation;
    private Allocation outAllocation;
    private int width;
    private int height;

    private ScriptIntrinsicBlur blurScript;

    public BlurProcessor(RenderScript rs) {
        this.rs = rs;
    }

    public void initialize(int width, int height) {
        blurScript = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
        blurScript.setRadius(7f); // Set blur radius. 25 is max

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }

        // Bitmap must have ARGB_8888 config for this type
        Type bitmapType = new Type.Builder(rs, Element.RGBA_8888(rs))
            .setX(width)
            .setY(height)
            .setMipmaps(false) // We are using MipmapControl.MIPMAP_NONE
    }
}

```

```

.create();

    // 创建输出分配
outAllocation = Allocation.createTyped(rs, bitmapType);

    // 创建与输出分配相同类型的输入分配
inAllocation = Allocation.createTyped(rs, bitmapType);
}

public void release() {

    if (blurScript != null) {
        blurScript.destroy();
        blurScript = null;
    }

    if (inAllocation != null) {
        inAllocation.destroy();
        inAllocation = null;
    }

    if (outAllocation != null) {
        outAllocation.destroy();
        outAllocation = null;
    }
}

public Bitmap process(Bitmap bitmap, boolean createNewBitmap) {
    if (bitmap.getWidth() != width || bitmap.getHeight() != height) {
        // 如有需要则抛出错误
        return null;
    }

    // 将数据从位图复制到输入分配
    inAllocation.copyFrom(bitmap);

    // 设置模糊脚本的输入
    blurScript.setInput(inAllocation);

    // 处理并将数据设置到输出分配
    blurScript.forEach(outAllocation);

    if (createNewBitmap) {
        Bitmap returnVal = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        outAllocation.copyTo(returnVal);
        return returnVal;
    }

    outAllocation.copyTo(bitmap);
    return bitmap;
}
}

```

```

.create();

    // Create output allocation
outAllocation = Allocation.createTyped(rs, bitmapType);

    // Create input allocation with same type as output allocation
inAllocation = Allocation.createTyped(rs, bitmapType);
}

public void release() {

    if (blurScript != null) {
        blurScript.destroy();
        blurScript = null;
    }

    if (inAllocation != null) {
        inAllocation.destroy();
        inAllocation = null;
    }

    if (outAllocation != null) {
        outAllocation.destroy();
        outAllocation = null;
    }
}

public Bitmap process(Bitmap bitmap, boolean createNewBitmap) {
    if (bitmap.getWidth() != width || bitmap.getHeight() != height) {
        // Throw error if required
        return null;
    }

    // Copy data from bitmap to input allocations
    inAllocation.copyFrom(bitmap);

    // Set input for blur script
    blurScript.setInput(inAllocation);

    // process and set data to the output allocation
    blurScript.forEach(outAllocation);

    if (createNewBitmap) {
        Bitmap returnVal = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        outAllocation.copyTo(returnVal);
        return returnVal;
    }

    outAllocation.copyTo(bitmap);
    return bitmap;
}
}

```

每个脚本都有一个处理数据的内核，通常通过forEach方法调用。

```

public class BlurActivity extends AppCompatActivity {
    private BlurProcessor blurProcessor;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // 设置布局和其他内容
    }
}

```

Each script has a kernel which processes the data and it is generally invoked via **forEach** method.

```

public class BlurActivity extends AppCompatActivity {
    private BlurProcessor blurProcessor;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // setup layout and other stuff
    }
}

```

```
blurProcessor = new BlurProcessor(RenderScript.create(getApplicationContext()));
}

private void loadImage(String path) {
    // 加载图像到位图
    Bitmap bitmap = loadBitmapFromPath(path);

    // 初始化该位图的处理器
    blurProcessor.release();
    blurProcessor.initialize(bitmap.getWidth(), bitmap.getHeight());

    // 模糊图像
    Bitmap blurImage = blurProcessor.process(bitmap, true); // 如果不想创建新位图, newBitmap参数设为false
}
}
```

这里的示例到此结束。建议在后台线程中进行处理。

```
blurProcessor = new BlurProcessor(RenderScript.create(getApplicationContext()));
}

private void loadImage(String path) {
    // Load image to bitmap
    Bitmap bitmap = loadBitmapFromPath(path);

    // Initialize processor for this bitmap
    blurProcessor.release();
    blurProcessor.initialize(bitmap.getWidth(), bitmap.getHeight());

    // Blur image
    Bitmap blurImage = blurProcessor.process(bitmap, true); // Use newBitmap as false if you
    don't want to create a new bitmap
}
}
```

This concluded the example here. It is advised to do the processing in a background thread.

第143章：Fresco

Fresco 是一个强大的系统，用于在安卓应用中显示图像。

在安卓4.x及更低版本中，Fresco将图像放置在安卓内存的一个特殊区域（称为ashmem）。这使得你的应用运行更快——并且大大减少了令人头疼的OutOfMemoryError错误的发生频率。

Fresco还支持JPEG的流式传输。

第143.1节：Fresco入门

首先，将Fresco添加到你的build.gradle中，如备注部分所示：

如果你需要额外的功能，比如动画GIF或WebP支持，你还需要添加相应的[Fresco artifacts](#)。

Fresco需要初始化。你只需初始化一次，因此将初始化放在你的Application中是个好主意。示例如下：

```
公共类MyApplication extendsApplication {  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        Fresco.initialize(this);  
    }  
}
```

如果你想从服务器加载远程图像，应用需要网络权限。只需将其添加到你的AndroidManifest.xml中：

```
<uses-permission android:name="android.permission.INTERNET" />
```

然后，在你的XML布局中添加一个SimpleDraweeView。Fresco不支持图像尺寸使用wrap_content，因为你可能有多个不同尺寸的图像（占位图、错误图、实际图等）。

所以你可以添加一个具有固定尺寸（或match_parent）的SimpleDraweeView：

```
<com.facebook.drawee.view.SimpleDraweeView  
    android:id="@+id/my_image_view"  
    android:layout_width="120dp"  
    android:layout_height="120dp"  
    fresco:placeholderImage="@drawable/placeholder" />
```

或者为你的图像提供一个宽高比：

```
<com.facebook.drawee.view.SimpleDraweeView  
    android:id="@+id/my_image_view"  
    android:layout_width="120dp"  
    android:layout_height="wrap_content"  
    fresco:viewAspectRatio="1.33"  
    fresco:placeholderImage="@drawable/placeholder" />
```

最后，你可以在Java中设置你的图像URI：

```
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
```

Chapter 143: Fresco

Fresco is a powerful system for displaying images in Android applications.

In Android 4.x and lower, Fresco puts images in a special region of **Android memory** (called ashmem). This lets your application run faster - and suffer the dreaded OutOfMemoryError much less often.

Fresco also supports streaming of JPEGs.

Section 143.1: Getting Started with Fresco

First, add Fresco to your build.gradle as shown in the Remarks section:

If you need additional features, like animated GIF or WebP support, you have to add the corresponding [Fresco artifacts](#) as well.

Fresco needs to be initialized. You should only do this 1 time, so placing the initialization in your Application is a good idea. An example for this would be:

```
public class MyApplication extends Application {  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        Fresco.initialize(this);  
    }  
}
```

If you want to load remote images from a server, your app needs the internet permission. Simply add it to your AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Then, add a SimpleDraweeView to your XML layout. Fresco does not support wrap_content for image dimensions since you might have multiple images with different dimensions (placeholder image, error image, actual image, ...).

So you can either add a SimpleDraweeView with fixed dimensions (or match_parent):

```
<com.facebook.drawee.view.SimpleDraweeView  
    android:id="@+id/my_image_view"  
    android:layout_width="120dp"  
    android:layout_height="120dp"  
    fresco:placeholderImage="@drawable/placeholder" />
```

Or supply an aspect ratio for your image:

```
<com.facebook.drawee.view.SimpleDraweeView  
    android:id="@+id/my_image_view"  
    android:layout_width="120dp"  
    android:layout_height="wrap_content"  
    fresco:viewAspectRatio="1.33"  
    fresco:placeholderImage="@drawable/placeholder" />
```

Finally, you can set your image URI in Java:

```
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
```

```
draweeView.setImageURI("http://yourdomain.com/yourimage.jpg");
```

就是这样！在网络图片加载完成之前，你应该能看到占位符图片。

第143.2节：使用OkHttp 3与Fresco

首先，除了正常的Fresco Gradle依赖外，你还需要在你的
build.gradle文件中添加：

```
compile "com.facebook.fresco:imagepipeline-okhttp3:1.2.0" // 或者更高版本。
```

当你初始化Fresco（通常在你自定义的Application实现中）时，现在可以指定你的OkHttp
客户端：

```
OkHttpClient okHttpClient = new OkHttpClient(); // 自行构建OkHttpClient.
```

```
Context context = ... // 你的应用程序上下文。  
ImagePipelineConfig config = OkHttpImagePipelineConfigFactory  
    .newBuilder(context, okHttpClient)  
    .build();  
Fresco.initialize(context, config);
```

第143.3节：使用DraweeController通过Fresco进 行JPEG流式传输

本示例假设您已经将Fresco添加到您的应用中（参见此示例）：

```
SimpleDraweeView img = new SimpleDraweeView(context);  
ImageRequest request = ImageRequestBuilder  
    .newBuilderWithSource(Uri.parse("http://example.com/image.png"))  
    .setProgressiveRenderingEnabled(true) // 这就是关键所在。  
    .build();  
  
DraweeController controller = Fresco.newDraweeControllerBuilder()  
    .setImageRequest(request)  
    .setOldController(img.getController()) // 从我们的  
    SimpleDraweeView获取当前控制器。  
    .build();  
  
img.setController(controller); // 将新控制器设置到SimpleDraweeView以启用  
渐进式JPEG。
```

```
draweeView.setImageURI("http://yourdomain.com/yourimage.jpg");
```

That's it! You should see your placeholder drawable until the network image has been fetched.

Section 143.2: Using OkHttp 3 with Fresco

First, in addition to the normal Fresco Gradle dependency, you have to add the OkHttp 3 dependency to your
build.gradle:

```
compile "com.facebook.fresco:imagepipeline-okhttp3:1.2.0" // Or a newer version.
```

When you initialize Fresco (usually in your custom Application implementation), you can now specify your OkHttp
client:

```
OkHttpClient okHttpClient = new OkHttpClient(); // Build on your own OkHttpClient.  
  
Context context = ... // Your Application context.  
ImagePipelineConfig config = OkHttpImagePipelineConfigFactory  
    .newBuilder(context, okHttpClient)  
    .build();  
Fresco.initialize(context, config);
```

Section 143.3: JPEG Streaming with Fresco using DraweeController

This example assumes that you have already added Fresco to your app (see this example):

```
SimpleDraweeView img = new SimpleDraweeView(context);  
ImageRequest request = ImageRequestBuilder  
    .newBuilderWithSource(Uri.parse("http://example.com/image.png"))  
    .setProgressiveRenderingEnabled(true) // This is where the magic happens.  
    .build();  
  
DraweeController controller = Fresco.newDraweeControllerBuilder()  
    .setImageRequest(request)  
    .setOldController(img.getController()) // Get the current controller from our  
    SimpleDraweeView.  
    .build();  
  
img.setController(controller); // Set the new controller to the SimpleDraweeView to enable  
progressive JPEGs.
```

第144章：下拉刷新

第144.1节：如何向您的应用添加下拉刷新

确保在应用的build.gradle文件的dependencies部分添加以下依赖：

```
compile 'com.android.support:support-core-ui:24.2.0'
```

然后在布局中添加SwipeRefreshLayout：

```
<android.support.v4.widget.SwipeRefreshLayout  
    android:id="@+id/swipe_refresh_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
    <!-- 在此放置你的视图 -->  
  
</android.support.v4.widget.SwipeRefreshLayout>
```

最后实现SwipeRefreshLayout.OnRefreshListener监听器。

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);  
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {  
    @Override  
    public void onRefresh() {  
        // 你的代码  
    }  
});
```

第144.2节：使用RecyclerView的下拉刷新

要添加一个带有Swipe To Refresh布局的RecyclerView，请在您的Activity/Fragment布局文件中添加以下内容：

```
<android.support.v4.widget.SwipeRefreshLayout  
    android:id="@+id/refresh_layout"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_behavior="@string/appbar_scrolling_view_behavior">  
  
    <android.support.v7.widget.RecyclerView  
        android:id="@+id/recycler_view"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical"  
        android:scrollbars="vertical" />  
  
</android.support.v4.widget.SwipeRefreshLayout>
```

在您的Activity/Fragment中添加以下代码以初始化SwipeRefreshLayout：

```
SwipeRefreshLayout mSwipeRefreshLayout = (SwipeRefreshLayout)  
findViewById(R.id.refresh_layout);  
mSwipeRefreshLayout.setColorSchemeResources(R.color.green_bg,  
    android.R.color.holo_green_light,  
    android.R.color.holo_orange_light,  
    android.R.color.holo_red_light);
```

Chapter 144: Swipe to Refresh

Section 144.1: How to add Swipe-to-Refresh To your app

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Then add the SwipeRefreshLayout in your layout:

```
<android.support.v4.widget.SwipeRefreshLayout  
    android:id="@+id/swipe_refresh_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
    <!-- place your view here -->  
  
</android.support.v4.widget.SwipeRefreshLayout>
```

Finally implement the SwipeRefreshLayout.OnRefreshListener listener.

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);  
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {  
    @Override  
    public void onRefresh() {  
        // your code  
    }  
});
```

Section 144.2: Swipe To Refresh with RecyclerView

To add a **Swipe To Refresh** layout with a **RecyclerView** add the following to your Activity/Fragment layout file:

```
<android.support.v4.widget.SwipeRefreshLayout  
    android:id="@+id/refresh_layout"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_behavior="@string/appbar_scrolling_view_behavior">  
  
    <android.support.v7.widget.RecyclerView  
        android:id="@+id/recycler_view"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical"  
        android:scrollbars="vertical" />  
  
</android.support.v4.widget.SwipeRefreshLayout>
```

In your Activity/Fragment add the following to initialize the **SwipeRefreshLayout**:

```
SwipeRefreshLayout mSwipeRefreshLayout = (SwipeRefreshLayout)  
findViewById(R.id.refresh_layout);  
mSwipeRefreshLayout.setColorSchemeResources(R.color.green_bg,  
    android.R.color.holo_green_light,  
    android.R.color.holo_orange_light,  
    android.R.color.holo_red_light);
```

```
mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
    @Override  
    public void onRefresh() {  
        // 当刷新布局被滑动时执行代码  
    }  
});
```

```
mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
    @Override  
    public void onRefresh() {  
        // Execute code when refresh layout swiped  
    }  
});
```

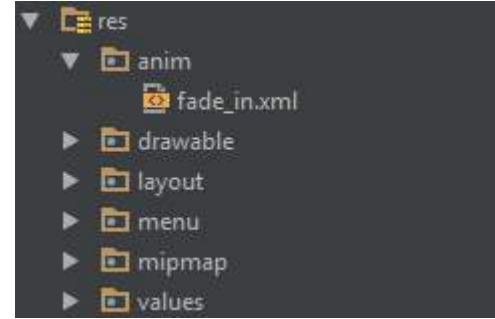
第145章：创建启动画面

第145.1节：带动画的启动画面

本示例展示了如何使用Android

步骤1：创建动画

在res目录下创建一个名为anim的新目录。右键点击该目录，创建一个新的Animation Resource文件，命名为fade_in.xml：



然后，将以下代码放入fade_in.xml文件中：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:fillAfter="true" >
    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />
</set>
```

步骤2：创建一个活动

使用Android Studio创建一个名为Splash的空活动。然后，将以下代码放入其中：

```
public class Splash extends AppCompatActivity {
    Animation anim;
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        imageView=(ImageView)findViewById(R.id.imageView2); // 声明一个imageView用于显示动画。
        anim = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in); // 创建动画。
        anim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                startActivity(new Intent(this,HomeActivity.class));
                // HomeActivity.class 是显示启动画面后要跳转的活动。
            }
        });
    }
}
```

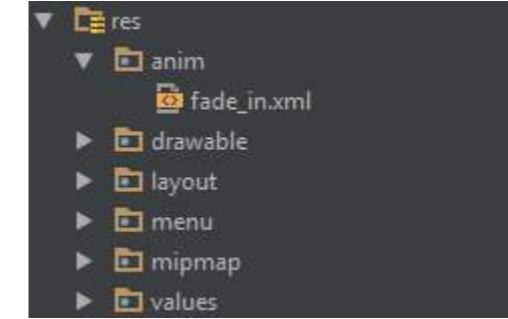
Chapter 145: Creating Splash screen

Section 145.1: Splash screen with animation

This example shows a simple but effective splash screen with animation that can be created by using Android Studio.

Step 1: Create an animation

Create a new directory named *anim* in the *res* directory. Right-click it and create a new *Animation Resource* file named *fade_in.xml*:



Then, put the following code into the *fade_in.xml* file:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:fillAfter="true" >
    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />
</set>
```

Step 2: Create an activity

Create an *empty activity* using Android Studio named *Splash*. Then, put the following code into it:

```
public class Splash extends AppCompatActivity {
    Animation anim;
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        imageView=(ImageView)findViewById(R.id.imageView2); // Declare an imageView to show the animation.
        anim = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in); // Create the animation.
        anim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                startActivity(new Intent(this,HomeActivity.class));
                // HomeActivity.class is the activity to go after showing the splash screen.
            }
        });
    }
}
```

```

@Override
public void onAnimationRepeat(Animation animation) {
}
);
imageView.startAnimation(anim);
}
}

```

接下来，将以下代码放入布局文件：

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_splash"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="your_packagename"
    android:orientation="vertical"
    android:background="@android:color/white">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/imageView2"
        android:layout_weight="1"
        android:src="@drawable/Your_logo_or_image" />
</LinearLayout>

```

步骤3：替换默认启动器

通过向AndroidManifest文件添加以下代码，将你的Splash活动变成启动器：

```

<activity
    android:name=".Splash"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

然后，通过从AndroidManifest文件中删除以下代码，移除默认启动器活动：

```

<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>

```

第145.2节：基础启动画面

启动画面就像其他任何活动一样，但它可以在后台处理你所有的启动需求。示例：

清单：

```

@Override
public void onAnimationRepeat(Animation animation) {
}
);
imageView.startAnimation(anim);
}
}

```

Next, put the following code into the layout file:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_splash"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="your_packagename"
    android:orientation="vertical"
    android:background="@android:color/white">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/imageView2"
        android:layout_weight="1"
        android:src="@drawable/Your_logo_or_image" />
</LinearLayout>

```

Step 3: Replace the default launcher

Turn your Splash activity into a launcher by adding the following code to the *AndroidManifest* file:

```

<activity
    android:name=".Splash"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

Then, remove the default launcher activity by removing the following code from the *AndroidManifest* file:

```

<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>

```

Section 145.2: A basic splash screen

A splash screen is just like any other activity, but it can handle all of your startup-needs in the background. Example:

Manifest:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Splash"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>

```

现在我们的启动屏幕将作为第一个活动被调用。

这是一个示例启动屏幕，同时处理一些关键的应用元素：

```

public class Splash extends Activity{

    public final int SPLASH_DISPLAY_LENGTH = 3000;

    private void checkPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WAKE_LOCK) != PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this, Manifest.permission.INTERNET) != PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_NETWORK_STATE) != PackageManager.PERMISSION_GRANTED) {//可以根据需求添加更多权限

            ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.WAKE_LOCK,
                    Manifest.permission.INTERNET,
                    Manifest.permission.ACCESS_NETWORK_STATE},
                123);
        }
    }

    @Override
    protected void onCreate(Bundle sis){
        super.onCreate(sis);
        //设置内容视图。XML 文件中可以只包含一张图片，比如徽标或应用图标
        setContentView(R.layout.splash);

        //我们希望显示启动画面几秒钟，然后自动跳转
    }
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Splash"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>

```

Now our splash-screen will be called as the first activity.

Here is an example splashscreen that also handles some critical app elements:

```

public class Splash extends Activity{

    public final int SPLASH_DISPLAY_LENGTH = 3000;

    private void checkPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WAKE_LOCK) != PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this, Manifest.permission.INTERNET) != PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_NETWORK_STATE) != PackageManager.PERMISSION_GRANTED) {//Can add more as per requirement

            ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.WAKE_LOCK,
                    Manifest.permission.INTERNET,
                    Manifest.permission.ACCESS_NETWORK_STATE},
                123);
        }
    }

    @Override
    protected void onCreate(Bundle sis){
        super.onCreate(sis);
        //set the content view. The XML file can contain nothing but an image, such as a logo or the
        //app icon
        setContentView(R.layout.splash);

        //we want to display the splash screen for a few seconds before it automatically
    }
}

```

```

//消失并加载游戏。所以我们创建一个线程：
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {

        //请求权限。注意：复制此代码和清单文件会导致应用崩溃，因为请求的权限未在清单中定义。
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            checkPermission();
        }
        String lang = [加载或确定系统语言，若不可用则设置为默认。]

        Locale locale = new Locale(lang);
        Locale.setDefault(locale);
        Configuration config = new Configuration();
        config.locale = locale;
        Splash.this.getResources().updateConfiguration(config,
            Splash.this.getResources().getDisplayMetrics()) ;

        //三秒后，将执行所有这些代码。
        //因此，我们接下来要重定向到主活动
        Intent mainIntent = new Intent(Splash.this, MainActivity.class);
        Splash.this.startActivity(mainIntent);

        //然后我们结束这个类。因为不再需要，释放它
        Splash.this.finish();
    }
}, SPLASH_DISPLAY_LENGTH);

}

public void onPause(){
    super.onPause();
    finish();
}
}

```

```

//disappears and loads the game. So we create a thread:
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {

        //request permissions. NOTE: Copying this and the manifest will cause the app to
        crash as the permissions requested aren't defined in the manifest.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            checkPermission();
        }
        String lang = [load or determine the system language and set to default if it
        isn't available.]
        Locale locale = new Locale(lang);
        Locale.setDefault(locale);
        Configuration config = new Configuration();
        config.locale = locale;
        Splash.this.getResources().updateConfiguration(config,
            Splash.this.getResources().getDisplayMetrics()) ;

        //after three seconds, it will execute all of this code.
        //as such, we then want to redirect to the master-activity
        Intent mainIntent = new Intent(Splash.this, MainActivity.class);
        Splash.this.startActivity(mainIntent);

        //then we finish this class. Dispose of it as it is longer needed
        Splash.this.finish();
    }
}, SPLASH_DISPLAY_LENGTH);

}

public void onPause(){
    super.onPause();
    finish();
}
}

```

第146章：IntentService

第146.1节：创建IntentService

要创建一个IntentService，需创建一个继承IntentService的类，并在其中重写onHandleIntent方法：

```
package com.example.myapp;
public class MyIntentService extends IntentService {
    @Override
    protected void onHandleIntent(Intent workIntent) {
        //根据workIntent的内容在后台执行某些操作。
    }
}
```

第146.2节：基本的IntentService示例

抽象类IntentService是服务的基类，服务在后台运行，没有任何用户界面。因此，为了更新UI，我们必须使用接收器，该接收器可以是BroadcastReceiver或ResultReceiver：

- 如果您的服务需要与多个想要监听通信的组件进行通信，应使用BroadcastReceiver。
- 如果您的服务只需要与父应用程序（即

您的应用程序）通信，则应使用ResultReceiver。

在IntentService中，有一个关键方法onHandleIntent()，我们将在其中执行所有操作，例如准备通知、创建闹钟等。

如果您想使用自己的IntentService，必须按如下方式继承它：

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: 在此编写您的代码。
    }
}
```

调用/启动该活动可以按如下方式进行：

```
Intent i = new Intent(this, YourIntentService.class);
startService(i); // 用于服务。
startActivity(i); // 用于活动；暂时忽略此行。
```

类似于任何活动，你可以通过如下方式传递额外信息，比如bundle数据：

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo", "bar");
startService(passDataIntent);
```

现在假设我们向YourIntentService类传递了一些数据。基于这些数据，可以执行以下操作：

Chapter 146: IntentService

Section 146.1: Creating an IntentService

To create an IntentService, create a class which extends IntentService, and within it, a method which overrides onHandleIntent:

```
package com.example.myapp;
public class MyIntentService extends IntentService {
    @Override
    protected void onHandleIntent (Intent workIntent) {
        //Do something in the background, based on the contents of workIntent.
    }
}
```

Section 146.2: Basic IntentService Example

The abstract class IntentService is a base class for services, which run in the background without any user interface. Therefore, in order to update the UI, we have to make use of a receiver, which may be either a BroadcastReceiver or a ResultReceiver:

- A BroadcastReceiver should be used if your service needs to communicate with multiple components that want to listen for communication.
- A ResultReceiver: should be used if your service needs to communicate with only the parent application (i.e. your application).

Within the IntentService, we have one key method, onHandleIntent(), in which we will do all actions, for example, preparing notifications, creating alarms, etc.

If you want to use your own IntentService, you have to extend it as follows:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

Calling/starting the activity can be done as follows:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i); // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Similar to any activity, you can pass extra information such as bundle data to it as follows:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo", "bar");
startService(passDataIntent);
```

Now assume that we passed some data to the YourIntentService class. Based on this data, an action can be

执行如下操作：

```
public class YourIntentService extends IntentService {  
    private String activityValue="bar";  
    String retrievedValue=intent.getStringExtra("foo");  
  
    public YourIntentService () {  
        super("YourIntentService ");  
    }  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        if(retrievedValue.equals(activityValue)){  
            // 向 foo 发送通知。  
        } else {  
            // 获取数据失败。  
        }  
    }  
}
```

上面的代码还展示了如何在 `OnHandleIntent()` 方法中处理约束。

第146.3节：示例 Intent 服务

下面是一个假装在后台加载图像的 `IntentService` 示例。实现一个 `IntentService` 所需做的就是提供一个调用 `super(String)` 构造函数的构造器，并且需要实现 `onHandleIntent(Intent)` 方法。

```
public class ImageLoaderIntentService extends IntentService {  
  
    public static final String IMAGE_URL = "url";  
  
    /**  
     * 定义一个构造函数并调用 super(String) 构造函数，以便为工作线程命名——如果你想调试并知道该服务执行其任务的线程  
     * 名称，这一点非常重要。  
     */  
    public ImageLoaderIntentService() {  
        super("Example");  
    }  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        // 这里是你执行所有逻辑的地方——这段代码在后台线程中执行  
  
        String imageUrl = intent.getStringExtra(IMAGE_URL);  
  
        if (!TextUtils.isEmpty(imageUrl)) {  
            Drawable image = HttpUtils.loadImage(imageUrl); // HttpUtils 是示例中虚构的工具类  
        }  
  
        // 现在将你的 Drawable 发送回 UI，以便你可以使用它——有很多方法可以实现这一点，  
        // 但这些方法超出了本示例的范围  
    }  
}
```

为了启动一个 `IntentService`，你需要向它发送一个 `Intent`。你可以从一个 `Activity` 中做到这一点，例如。当然，你并不限于此。下面是一个如何从 `Activity` 类调用你的新 `Service` 的示例。

performed as follows:

```
public class YourIntentService extends IntentService {  
    private String activityValue="bar";  
    String retrievedValue=intent.getStringExtra("foo");  
  
    public YourIntentService () {  
        super("YourIntentService ");  
    }  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        if(retrievedValue.equals(activityValue)){  
            // Send the notification to foo.  
        } else {  
            // Retrieving data failed.  
        }  
    }  
}
```

The code above also shows how to handle constraints in the `OnHandleIntent()` method.

Section 146.3: Sample Intent Service

Here is an example of an `IntentService` that pretends to load images in the background. All you need to do to implement an `IntentService` is to provide a constructor that calls the `super(String)` constructor, and you need to implement the `onHandleIntent(Intent)` method.

```
public class ImageLoaderIntentService extends IntentService {  
  
    public static final String IMAGE_URL = "url";  
  
    /**  
     * Define a constructor and call the super(String) constructor, in order to name the worker  
     * thread - this is important if you want to debug and know the name of the thread upon  
     * which this Service is operating its jobs.  
     */  
    public ImageLoaderIntentService() {  
        super("Example");  
    }  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        // This is where you do all your logic - this code is executed on a background thread  
  
        String imageUrl = intent.getStringExtra(IMAGE_URL);  
  
        if (!TextUtils.isEmpty(imageUrl)) {  
            Drawable image = HttpUtils.loadImage(imageUrl); // HttpUtils is made-up for the example  
        }  
  
        // Send your drawable back to the UI now, so that you can use it - there are many ways  
        // to achieve this, but they are out of reach for this example  
    }  
}
```

In order to start an `IntentService`, you need to send an `Intent` to it. You can do so from an `Activity`, for an example. Of course, you're not limited to that. Here is an example of how you would summon your new `Service` from an `Activity` class.

```
Intent serviceIntent = new Intent(this, ImageLoaderIntentService.class); // 如果你的类是 Context (例如 Activity、另一个 Service 等) , 你可以使用 'this' 作为第一个参数, 否则需要以其他方式提供 Context
```

```
serviceIntent.putExtra(IMAGE_URL, "http://www.example-site.org/some/path/to/an/image");  
startService(serviceIntent); // 如果第一行没有使用 'this', 你还需要在这里调用 Context 对象的 startService(Intent)
```

IntentService 会顺序处理来自其 Intent 的数据，因此你可以发送多个 Intent，而不必担心它们会相互冲突。一次只处理一个 Intent，其他的会排队等待。

当所有任务完成后，IntentService 会自动关闭自身。

```
Intent serviceIntent = new Intent(this, ImageLoaderIntentService.class); // you can use 'this' as the first parameter if your class is a Context (i.e. an Activity, another Service, etc.), otherwise, supply the context differently  
serviceIntent.putExtra(IMAGE_URL, "http://www.example-site.org/some/path/to/an/image");  
startService(serviceIntent); // if you are not using 'this' in the first line, you also have to put the call to the Context object before startService(Intent) here
```

The IntentService processes the data from its Intents sequentially, so that you can send multiple Intents without worrying whether they will collide with each other. Only one Intent at a time is processed, the rest go in a queue. When all the jobs are complete, the IntentService will shut itself down automatically.

第147章：隐式意图

参数	详细信息
o	意图
动作	字符串: 意图动作, 例如 ACTION_VIEW。
uri	Uri: 意图数据URI。
packageContext	上下文: 实现此类的应用程序包的上下文。
cls	类: 用于意图的组件类。

第147.1节：隐式和显式意图

显式意图用于启动同一应用包内的活动或服务。在这种情况下，明确指定了目标类的名称：

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

然而，隐式意图会发送到系统中，供用户设备上安装的任何能够处理该意图的应用使用。这用于不同应用之间共享信息。

```
Intent intent = new Intent("com.stackoverflow.example.VIEW");

//我们需要检查是否有应用安装能够处理此意图
if (getPackageManager().resolveActivity(intent, 0) != null){
    startActivity(intent);
} else{
    //处理错误
}
```

有关差异的更多详细信息，请参见Android开发者文档：[意图解析 \(Intent Resolution\)](#)

第147.2节：隐式意图

隐式意图不指定具体组件，而是声明要执行的一般操作，这允许来自其他应用的组件来处理它。

例如，如果您想在地图上向用户显示一个位置，可以使用隐式意图请求另一个有能力的应用在地图上显示指定的位置。

示例：

```
// 使用字符串创建文本消息
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// 验证该意图是否能解析到一个活动
if (sendIntent.resolveActivity(getApplicationContext()) != null) {
    startActivity(sendIntent);
}
```

Chapter 147: Implicit Intents

Parameters	Details
o	Intent
action	String: The Intent action, such as ACTION_VIEW.
uri	Uri: The Intent data URI.
packageContext	Context: A Context of the application package implementing this class.
cls	Class: The component class that is to be used for the intent.

Section 147.1: Implicit and Explicit Intents

An explicit intent is used for starting an activity or service within the same application package. In this case the name of the intended class is explicitly mentioned:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

However, an implicit intent is sent across the system for any application installed on the user's device that can handle that intent. This is used to share information between different applications.

```
Intent intent = new Intent("com.stackoverflow.example.VIEW");

//We need to check to see if there is an application installed that can handle this intent
if (getPackageManager().resolveActivity(intent, 0) != null){
    startActivity(intent);
} else{
    //Handle error
}
```

More details on the differences can be found in the Android Developer docs here: [Intent Resolution](#)

Section 147.2: Implicit Intents

[Implicit](#) intents do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

Example:

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getApplicationContext()) != null) {
    startActivity(sendIntent);
}
```

第148章：发布到Play商店

第148.1节：最简应用提交指南

要求：

- 开发者账号
- 已构建并使用非调试密钥签名的apk
- 一个没有内购的免费应用
- 无Firebase云消息或游戏服务

1. 前往<https://play.google.com/apps/publish/>

1a) 如果没有开发者账号，请创建一个

2. 点击按钮创建新应用

3. 点击提交APK

4. 填写表单中的所有必填字段，包括将在Play商店展示的一些资源（见下图）

5. 确认无误后点击发布应用按钮



Chapter 148: Publish to Play Store

Section 148.1: Minimal app submission guide

Requirements:

- A developer account
- An apk already built and signed with a non-debug key
- A free app that doesn't have in-app billing
- no Firebase Cloud Messaging or Game Services

1. Head to <https://play.google.com/apps/publish/>

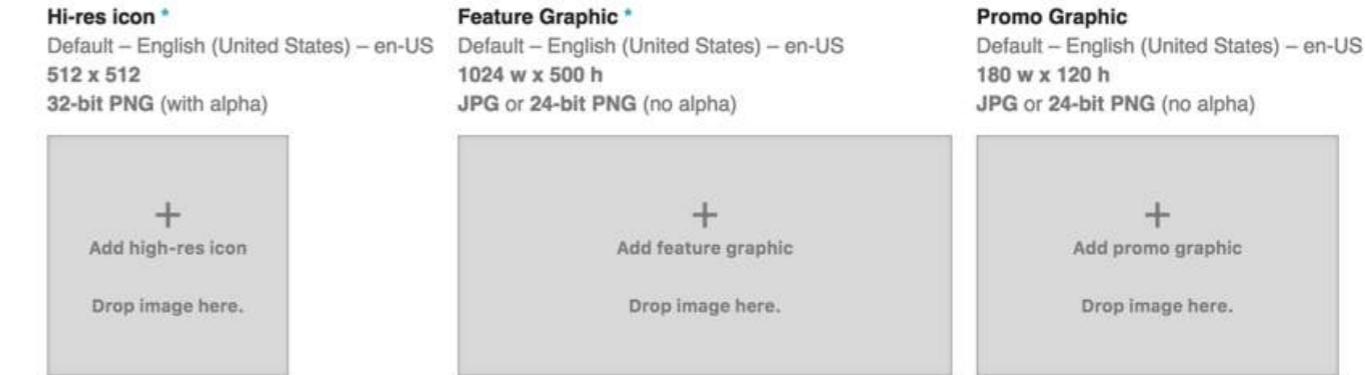
1a) Create your developer account if you do not have one

2. Click button Create new Application

3. Click submit APK

4. Fill in all required fields in the form, including some assets that will be displayed on the Play Store (see image below)

5. When satisfied hit Publish app button



UPLOAD NEW APK TO PRODUCTION

com.example.demo.app		
Version code 8	Version name 1.2.1	Size 4.87 MB

APK details Hide

Differences from the previous version are highlighted

Supported Android devices	10495 devices (105 added)
API levels	16+
Screen layouts	4 screen layouts ▾
Localizations	default language only
Features	1 feature (1 removed) ▾
Required permissions	6 permissions ▾
OpenGL ES versions	1.0+
OpenGL textures	all textures

Use expansion file ?

No expansion file
▼

查看更多关于签名的信息，请配置签名设置

UPLOAD NEW APK TO PRODUCTION

com.example.demo.app		
Version code 8	Version name 1.2.1	Size 4.87 MB

APK details Hide

Differences from the previous version are highlighted

Supported Android devices	10495 devices (105 added)
API levels	16+
Screen layouts	4 screen layouts ▾
Localizations	default language only
Features	1 feature (1 removed) ▾
Required permissions	6 permissions ▾
OpenGL ES versions	1.0+
OpenGL textures	all textures

Use expansion file ?

No expansion file
▼

See more about signing in Configure Signing Settings

第149章：通用图像加载器

第149.1节：基本用法

1. 加载图像，将其解码为位图，并在ImageView（或任何其他视图）中显示该位图实现了ImageAware接口：

```
ImageLoader.getInstance().displayImage(imageUri, imageView);
```

2. 加载一张图片，将其解码为位图，并将位图返回给回调函数：

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {  
    @Override  
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {  
        // 对位图执行你想做的任何操作。  
    }  
});
```

3. 加载图像，将其解码为位图并同步返回该位图：

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

第149.2节：初始化通用图片加载器

1. 在build.gradle文件中添加以下依赖：

```
compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
```

2. 在AndroidManifest.xml文件中添加以下权限：

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

3. 初始化通用图片加载器。此操作必须在首次使用之前完成：

```
ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)  
    // ...  
.build();  
ImageLoader.getInstance().init(config);
```

完整的配置选项可以在这里找到。[here](#)

Chapter 149: Universal Image Loader

Section 149.1: Basic usage

1. Load an image, decode it into a bitmap, and display the bitmap in an ImageView (or any other view which implements the ImageAware interface):

```
ImageLoader.getInstance().displayImage(imageUri, imageView);
```

2. Load an image, decode it into a bitmap, and return the bitmap to a callback:

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {  
    @Override  
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {  
        // Do whatever you want with the bitmap.  
    }  
});
```

3. Load an image, decode it into a bitmap and return the bitmap synchronously:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Section 149.2: Initialize Universal Image Loader

1. Add the following dependency to the build.gradle file:

```
compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
```

2. Add the following permissions to the AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

3. Initialize the Universal Image Loader. This must be done before the first usage:

```
ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)  
    // ...  
.build();  
ImageLoader.getInstance().init(config);
```

The full configuration options can be found [here](#).

第150章：图像压缩

第150.1节：如何在不改变尺寸的情况下压缩图像

从单例类获取压缩位图：

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
Bitmap bitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
imageView.setImageBitmap(bitmap);
```

ImageUtils.java：

```
public class ImageUtils {

    public static ImageUtils mInstant;

    public static ImageUtils getInstant(){
        if(mInstant==null){
            mInstant = new ImageUtils();
        }
        return mInstant;
    }

    public Bitmap getCompressedBitmap(String imagePath) {
        float maxHeight = 1920.0f;
        float maxWidth = 1080.0f;
        Bitmap scaledBitmap = null;
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        Bitmap bmp = BitmapFactory.decodeFile(imagePath, options);

        int actualHeight = options.outHeight;
        int actualWidth = options.outWidth;
        float imgRatio = (float) actualWidth / (float) actualHeight;
        float maxRatio = maxWidth / maxHeight;

        if (actualHeight > maxHeight || actualWidth > maxWidth) {
            if (imgRatio < maxRatio) {
                imgRatio = maxHeight / actualHeight;
                actualWidth = (int) (imgRatio * actualWidth);
                actualHeight = (int) maxHeight;
            } else if (imgRatio > maxRatio) {
                imgRatio = maxWidth / actualWidth;
                actualHeight = (int) (imgRatio * actualHeight);
                actualWidth = (int) maxWidth;
            } else {
                actualHeight = (int) maxHeight;
                actualWidth = (int) maxWidth;
            }
        }

        options.inSampleSize = calculateInSampleSize(options, actualWidth, actualHeight);
        options.inJustDecodeBounds = false;
        options.inDither = false;
        options.inPurgeable = true;
        options.inInputShareable = true;
        options.inTempStorage = new byte[16 * 1024];
    }
}
```

Chapter 150: Image Compression

Section 150.1: How to compress image without size change

Get **Compressed Bitmap** from Singleton class:

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
Bitmap bitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
imageView.setImageBitmap(bitmap);
```

ImageUtils.java:

```
public class ImageUtils {

    public static ImageUtils mInstant;

    public static ImageUtils getInstant(){
        if(mInstant==null){
            mInstant = new ImageUtils();
        }
        return mInstant;
    }

    public Bitmap getCompressedBitmap(String imagePath) {
        float maxHeight = 1920.0f;
        float maxWidth = 1080.0f;
        Bitmap scaledBitmap = null;
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        Bitmap bmp = BitmapFactory.decodeFile(imagePath, options);

        int actualHeight = options.outHeight;
        int actualWidth = options.outWidth;
        float imgRatio = (float) actualWidth / (float) actualHeight;
        float maxRatio = maxWidth / maxHeight;

        if (actualHeight > maxHeight || actualWidth > maxWidth) {
            if (imgRatio < maxRatio) {
                imgRatio = maxHeight / actualHeight;
                actualWidth = (int) (imgRatio * actualWidth);
                actualHeight = (int) maxHeight;
            } else if (imgRatio > maxRatio) {
                imgRatio = maxWidth / actualWidth;
                actualHeight = (int) (imgRatio * actualHeight);
                actualWidth = (int) maxWidth;
            } else {
                actualHeight = (int) maxHeight;
                actualWidth = (int) maxWidth;
            }
        }

        options.inSampleSize = calculateInSampleSize(options, actualWidth, actualHeight);
        options.inJustDecodeBounds = false;
        options.inDither = false;
        options.inPurgeable = true;
        options.inInputShareable = true;
        options.inTempStorage = new byte[16 * 1024];
    }
}
```

```

try {
    bmp = BitmapFactory.decodeFile(imagePath, options);
} catch (OutOfMemoryError exception) {
    exception.printStackTrace();
}

try {
    scaledBitmap = Bitmap.createBitmap(actualWidth, actualHeight, Bitmap.Config.ARGB_8888);
} catch (OutOfMemoryError exception) {
    exception.printStackTrace();
}

float ratioX = actualWidth / (float) options.outWidth;
float ratioY = actualHeight / (float) options.outHeight;
float middleX = actualWidth / 2.0f;
float middleY = actualHeight / 2.0f;

Matrix scaleMatrix = new Matrix();
scaleMatrix.setScale(ratioX, ratioY, middleX, middleY);

Canvas canvas = new Canvas(scaledBitmap);
canvas.setMatrix(scaleMatrix);
canvas.drawBitmap(bmp, middleX - bmp.getWidth() / 2, middleY - bmp.getHeight() / 2, new
Paint(Paint.FILTER_BITMAP_FLAG));

ExifInterface exif = null;
try {
    exif = new ExifInterface(imagePath);
    int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, 0);
    Matrix matrix = new Matrix();
    if (orientation == 6) {
        matrix.postRotate(90);
    } else if (orientation == 3) {
        matrix.postRotate(180);
    } else if (orientation == 8) {
        matrix.postRotate(270);
    }
}
scaledBitmap = Bitmap.createBitmap(scaledBitmap, 0, 0, scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix, true);
} catch (IOException e) {
e.printStackTrace();
}
ByteArrayOutputStream out = new ByteArrayOutputStream();
scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 85, out);

byte[] byteArray = out.toByteArray();

Bitmap updatedBitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);

return updatedBitmap;
}

private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int reqHeight) {
    final int height = options.outHeight;
    final int width = options.outWidth;
    int inSampleSize = 1;

    if (height > reqHeight || width > reqWidth) {
        final int heightRatio = Math.round((float) height / (float) reqHeight);
        final int widthRatio = Math.round((float) width / (float) reqWidth);
        inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
    }
}

```

```

try {
    bmp = BitmapFactory.decodeFile(imagePath, options);
} catch (OutOfMemoryError exception) {
    exception.printStackTrace();
}

try {
    scaledBitmap = Bitmap.createBitmap(actualWidth, actualHeight, Bitmap.Config.ARGB_8888);
} catch (OutOfMemoryError exception) {
    exception.printStackTrace();
}

float ratioX = actualWidth / (float) options.outWidth;
float ratioY = actualHeight / (float) options.outHeight;
float middleX = actualWidth / 2.0f;
float middleY = actualHeight / 2.0f;

Matrix scaleMatrix = new Matrix();
scaleMatrix.setScale(ratioX, ratioY, middleX, middleY);

Canvas canvas = new Canvas(scaledBitmap);
canvas.setMatrix(scaleMatrix);
canvas.drawBitmap(bmp, middleX - bmp.getWidth() / 2, middleY - bmp.getHeight() / 2, new
Paint(Paint.FILTER_BITMAP_FLAG));

ExifInterface exif = null;
try {
    exif = new ExifInterface(imagePath);
    int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, 0);
    Matrix matrix = new Matrix();
    if (orientation == 6) {
        matrix.postRotate(90);
    } else if (orientation == 3) {
        matrix.postRotate(180);
    } else if (orientation == 8) {
        matrix.postRotate(270);
    }
}
scaledBitmap = Bitmap.createBitmap(scaledBitmap, 0, 0, scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix, true);
} catch (IOException e) {
e.printStackTrace();
}
ByteArrayOutputStream out = new ByteArrayOutputStream();
scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 85, out);

byte[] byteArray = out.toByteArray();

Bitmap updatedBitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);

return updatedBitmap;
}

private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int reqHeight) {
    final int height = options.outHeight;
    final int width = options.outWidth;
    int inSampleSize = 1;

    if (height > reqHeight || width > reqWidth) {
        final int heightRatio = Math.round((float) height / (float) reqHeight);
        final int widthRatio = Math.round((float) width / (float) reqWidth);
        inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
    }
}

```

```

final float totalPixels = width * height;
final float totalReqPixelsCap = reqWidth * reqHeight * 2;

while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
    inSampleSize++;
}
return inSampleSize;
}

```

压缩后的Bitmap尺寸相同。

我是如何检查的？

```

Bitmap beforeBitmap = BitmapFactory.decodeFile("Your_Image_Path_Here");
Log.i("压缩前尺寸", beforeBitmap.getWidth()+"-"+beforeBitmap.getHeight());

Bitmap afterBitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
Log.i("压缩后尺寸", afterBitmap.getWidth() + " - " + afterBitmap.getHeight());

```

输出：

压缩前:尺寸:1080-1452
压缩后:尺寸:1080-1452

```

final float totalPixels = width * height;
final float totalReqPixelsCap = reqWidth * reqHeight * 2;

while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
    inSampleSize++;
}
return inSampleSize;
}

```

Dimensions are same after compressing Bitmap.

How did I checked ?

```

Bitmap beforeBitmap = BitmapFactory.decodeFile("Your_Image_Path_Here");
Log.i("Before Compress Dimension", beforeBitmap.getWidth()+"-"+beforeBitmap.getHeight());

Bitmap afterBitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
Log.i("After Compress Dimension", afterBitmap.getWidth() + " - " + afterBitmap.getHeight());

```

Output:

Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452

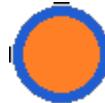
第151章：9-补丁图像

第151.1节：基本圆角

正确拉伸的关键在于顶部和左侧边框。

顶部边框控制水平拉伸，左侧边框控制垂直拉伸。

此示例创建了适用于吐司（Toast）的圆角。



图像中位于顶部边框以下和左侧边框右侧的部分将扩展以填充所有未使用的空间。

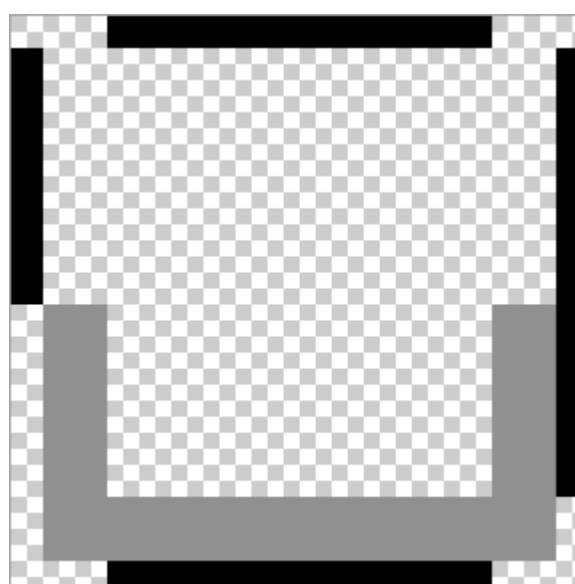
此示例将拉伸到所有尺寸组合，如下所示：



第151.2节：可选填充线

九补丁图像允许可选定义图像中的填充线。填充线是位于右侧和底部的线。

如果一个视图将9-patch图像设置为其背景，填充线用于定义视图内容的空间（例如，EditText中的文本输入）。如果未定义填充线，则使用左侧和顶部的线代替。



拉伸图像的内容区域看起来如下：

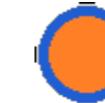
Chapter 151: 9-Patch Images

Section 151.1: Basic rounded corners

The key to correctly stretching is in the top and left border.

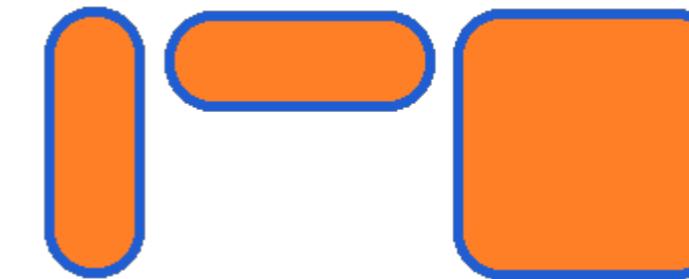
The top border controls horizontal stretching and the left border controls vertical stretching.

This example creates rounded corners suitable for a Toast.



The parts of the image that are below the *top border* and to the right of the *left border* will expand to fill all unused space.

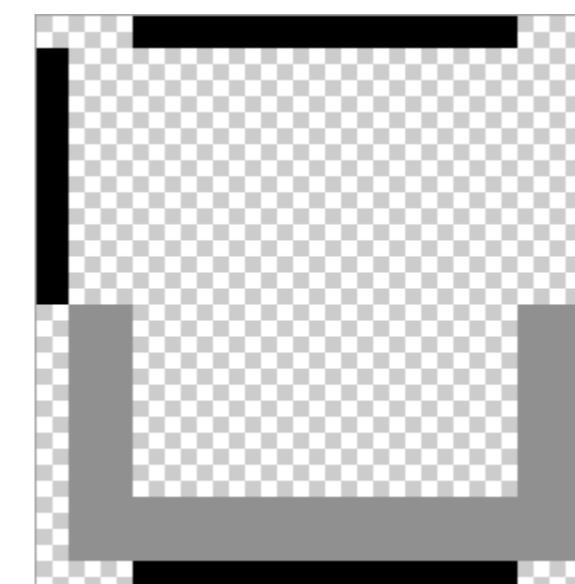
This example will stretch to all combinations of sizes, as shown below:



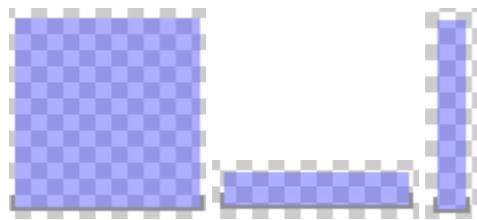
Section 151.2: Optional padding lines

Nine-patch images allow optional definition of the padding lines in the image. The padding lines are the lines on the right and at the bottom.

If a View sets the 9-patch image as its background, the padding lines are used to define the space for the View's content (e.g. the text input in an EditText). If the padding lines are not defined, the left and top lines are used instead.



The content area of the stretched image then looks like this:



第151.3节：基础下拉框

Spinner 可以根据您自己的样式需求使用九宫格进行重绘皮肤。

例如，参见此九宫格：



如您所见，它标记了3个极小的拉伸区域。

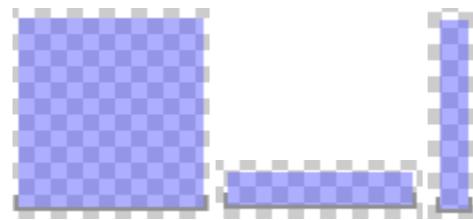
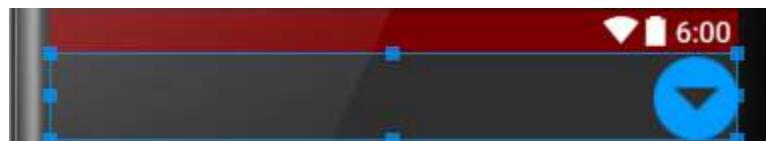
顶部边框仅标记了图标的左侧。这表示我希望可绘制对象的左侧（完全透明）填充Spinner视图，直到图标位置。

左边框在图标的顶部和底部标记了透明段。这表示顶部和底部都会扩展到Spinner视图的大小。这将使图标本身垂直居中。

使用没有九宫格元数据的图像：



使用带有九宫格元数据的图像：



Section 151.3: Basic spinner

The Spinner can be reskinned according to your own style requirements using a Nine Patch.

As an example, see this Nine Patch:



As you can see, it has 3 extremely small areas of stretching marked.

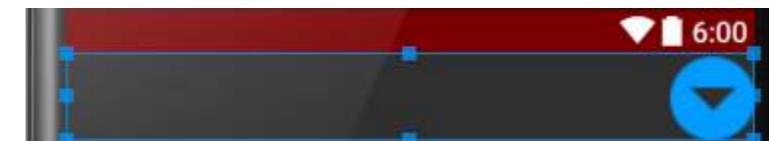
The top border has only left of the icon marked. That indicates that I want the left side (complete transparency) of the drawable to fill the Spinner view until the icon is reached.

The left border has marked transparent segments at the top and bottom of the icon marked. That indicates that both the top and the bottom will expand to the size of the Spinner view. This will leave the icon itself centered vertically.

Using the image without Nine Patch metadata:



Using the image with Nine Patch metadata:



第152章：电子邮件验证

第152.1节：电子邮件地址验证

添加以下方法以检查电子邮件地址是否有效：

```
private boolean isValidEmailId(String email){  
    return Pattern.compile("^(\\w+\\.\\w+|([a-zA-Z]{1}|[\\w-]{2,})@"  
        + "[0-1]?[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\."  
        + "[0-1]?[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[a-zA-Z]+[\\w-]+\\.[a-zA-Z]{2,4}$").matcher(email).matches();  
}
```

上述方法可以通过将EditText控件的文本转换为String来轻松验证：

```
if(isValidEmailId(edtEmailId.getText().toString().trim())){  
    Toast.makeText(getApplicationContext(), "有效的电子邮件地址。", Toast.LENGTH_SHORT).show();  
} else{  
    Toast.makeText(getApplicationContext(), "无效的电子邮件地址。", Toast.LENGTH_SHORT).show();  
}
```

第152.2节：使用Patterns进行电子邮件地址验证

```
if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){  
    Log.i("EmailCheck", "有效");  
}
```

Chapter 152: Email Validation

Section 152.1: Email address validation

Add the following method to check whether an email address is valid or not:

```
private boolean isValidEmailId(String email){  
    return Pattern.compile("^(\\w+\\.\\w+|([a-zA-Z]{1}|[\\w-]{2,})@"  
        + "[0-1]?[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\."  
        + "[0-1]?[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[0-9]{1,2}|2[0-5]|2[0-4][0-9])\\.(\\d-1?"  
        + "[a-zA-Z]+[\\w-]+\\.[a-zA-Z]{2,4}$").matcher(email).matches();  
}
```

The above method can easily be verified by converting the text of an EditText widget into a `String`:

```
if(isValidEmailId(edtEmailId.getText().toString().trim())){  
    Toast.makeText(getApplicationContext(), "Valid Email Address.", Toast.LENGTH_SHORT).show();  
} else{  
    Toast.makeText(getApplicationContext(), "InValid Email Address.", Toast.LENGTH_SHORT).show();  
}
```

Section 152.2: Email Address validation with using Patterns

```
if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){  
    Log.i("EmailCheck", "It is valid");  
}
```

第153章：底部弹出页

底部弹出页是从屏幕底部边缘滑出的页面。

第153.1节：快速设置

确保在应用的build.gradle文件的dependencies下添加以下依赖：

```
编译 'com.android.support:design:25.3.1'
```

然后你可以使用以下选项来使用底部弹出层：

- BottomSheetBehavior 用于 CoordinatorLayoutBottomSheetDialogDia
- log 是带有底部弹出层行为的对话框BottomSheetDialogFragment 是 Dialog
- gFragment 的扩展，创建一个 BottomSheetDialog 而不是标准对话框。

第153.2节：类似谷歌地图的BottomSheetBehavior

版本 ≥ 2.1.x

此示例依赖于支持库23.4.0及以上版本。

BottomSheetBehavior的特点包括：

- 1.两个带动画的工具栏，响应底部弹出层的移动。
- 2.一个FAB（悬浮操作按钮），当靠近“模态工具栏”（滑动时出现的工具栏）时会隐藏。
- 3.底部弹出层后面的背景图像，带有某种视差效果。
- 4.工具栏中的标题（TextView），当底部弹出层到达时显示。
- 5.通知状态栏可以将背景变为透明或全色。
- 6.带有“锚点”状态的自定义底部弹出层行为。

现在让我们逐一检查它们：

工具栏

当你在谷歌地图中打开该视图时，你会看到一个可以搜索的工具栏，我没有完全按照谷歌地图的做法，因为我想做得更通用一些。无论如何，这个ToolBar是在一个AppBarLayout中，当你开始拖动底部弹出层时它会隐藏，当底部弹出层达到COLLAPSED状态时它会再次出现。

要实现这一点，你需要：

- 创建一个Behavior并继承自AppBarLayout.ScrollingViewBehavior，重写layoutDependsOn和onDependentViewChanged方法。这样你就能监听到底部弹出层的移动。
- 创建一些方法，通过动画来隐藏和显示AppBarLayout/ToolBar。

这是我为第一个工具栏或操作栏所做的实现：

```
@Override
```

Chapter 153: Bottom Sheets

A bottom sheet is a sheet that slides up from the bottom edge of the screen.

Section 153.1: Quick Setup

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Then you can use the Bottom sheet using these options:

- BottomSheetBehavior to be used with CoordinatorLayout
- BottomSheetDialog which is a dialog with a bottom sheet behavior
- BottomSheetDialogFragment which is an extension of DialogFragment, that creates a BottomSheetDialog instead of a standard dialog.

Section 153.2: BottomSheetBehavior like Google maps

Version ≥ 2.1.x

This example depends on Support Library 23.4.0.+.

BottomSheetBehavior is characterized by :

1. Two toolbars with animations that respond to the bottom sheet movements.
2. A FAB that hides when it is near to the "modal toolbar" (the one that appears when you are sliding up).
3. A backdrop image behind bottom sheet with some kind of parallax effect.
4. A Title (TextView) in Toolbar that appears when bottom sheet reach it.
5. The notification status bar can turn its background to transparent or full color.
6. A custom bottom sheet behavior with an "anchor" state.

Now let's check them one by one:

ToolBars

When you open that view in Google Maps, you can see a toolbar in where you can search, it's the only one that I'm not doing exactly like Google Maps, because I wanted to do it more generic. Anyway that Toolbar is inside an AppBarLayout and it got hidden when you start dragging the BottomSheet and it appears again when the BottomSheet reach the COLLAPSED state.

To achieve it you need to:

- create a Behavior and extend it from AppBarLayout.ScrollingViewBehavior
- override layoutDependsOn and onDependentViewChanged methods. Doing it you will listen for bottomSheet movements.
- create some methods to hide and unhide the AppBarLayout/ToolBar with animations.

This is how I did it for first toolbar or ActionBar:

```
@Override
```

```

public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                       View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //向上滑动
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
    BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
    BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
        @BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        });
    });
}

private void dismissAppBar(View child){
    hidden = true;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
    appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shortAnimTime));
    mToolbarAnimation.y(-(mChild.getHeight()+25)).start();
}

private void showAppBar(View child) {
    hidden = false;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =

```

```

public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                       View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
    BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
    BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
        @BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        });
    });
}

private void dismissAppBar(View child){
    hidden = true;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
    appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shortAnimTime));
    mToolbarAnimation.y(-(mChild.getHeight()+25)).start();
}

private void showAppBar(View child) {
    hidden = false;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =

```

```

appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_mediumAnimTime));
mToolbarAnimation.y(mInitialY).start();
}

```

[如果需要，我可以提供完整文件](#)

第二个工具栏或“模态”工具栏：

你必须重写相同的方法，但在这个中你需要处理更多的行为：

- 使用动画显示/隐藏工具栏
- 更改状态栏颜色/背景
- 在工具栏中显示/隐藏底部弹出框标题
- 关闭底部弹出框或将其发送到折叠状态

这部分代码稍微复杂一些，所以我会提供链接

悬浮操作按钮 (FAB)

这也是一个自定义行为，但继承自FloatingActionButton.Behavior。在onDependentViewChanged中，你需要观察何时达到“偏移量”或你想隐藏它的点。就我而言，我想在它接近第二个工具栏时隐藏它，所以我深入FAB的父布局（一个CoordinatorLayout）寻找包含工具栏的AppBarLayout，然后我使用工具栏的位置作为偏移量：

```

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, FloatingActionButton child, View dependency) {

    if (offset == 0)
        setOffsetValue(parent);

    if (dependency.getY() <= 0)
        return false;

    if (child.getY() <= (offset + child.getHeight()) && child.getVisibility() == View.VISIBLE)
        child.hide();
    else if (child.getY() > offset && child.getVisibility() != View.VISIBLE)
        child.show();

    return false;
}

```

[完整的自定义FAB行为链接](#)

带视差效果的BottomSheet背后的图像：

和其他一样，这是一个自定义行为，这个行为中唯一“复杂”的部分是那个小算法，它保持图像锚定在BottomSheet上，避免图像像默认视差效果那样塌陷：

```

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mYmultiplier == 0) {
        initValues(child, dependency);
}

```

```

appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_mediumAnimTime));
mToolbarAnimation.y(mInitialY).start();
}

```

[Here is the complete file if you need it](#)

The second Toolbar or "Modal" toolbar:

You have to override the same methods, but in this one you have to take care of more behaviors:

- show/hide the ToolBar with animations
- change status bar color/background
- show/hide the BottomSheet title in the ToolBar
- close the bottomSheet or send it to collapsed state

The code for this one is a little extensive, so I will let [the link](#)

The FAB

This is a Custom Behavior too, but extends from FloatingActionButton.Behavior. In onDependentViewChanged you have to look when it reaches the "offSet" or point in where you want to hide it. In my case I want to hide it when it's near to the second toolbar, so I dig into FAB parent (a CoordinatorLayout) looking for the AppBarLayout that contains the ToolBar, then I use the ToolBar position like OffSet:

```

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, FloatingActionButton child, View dependency) {

    if (offset == 0)
        setOffsetValue(parent);

    if (dependency.getY() <= 0)
        return false;

    if (child.getY() <= (offset + child.getHeight()) && child.getVisibility() == View.VISIBLE)
        child.hide();
    else if (child.getY() > offset && child.getVisibility() != View.VISIBLE)
        child.show();

    return false;
}

```

[Complete Custom FAB Behavior link](#)

The image behind the BottomSheet with parallax effect:

Like the others, it's a custom behavior, the only "complicated" thing in this one is the little algorithm that keeps the image anchored to the BottomSheet and avoid the image collapse like default parallax effect:

```

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mYmultiplier == 0) {
        initValues(child, dependency);
}

```

```

    return true;
}

float dVerticalScroll = dependency.getY() - mPreviousY;
mPreviousY = dependency.getY();

//向上滑动
if (dVerticalScroll <= 0 && child.getY() <= 0) {
    child.setY(0);
    return true;
}

//向下移动
if (dVerticalScroll >= 0 && dependency.getY() <= mImageHeight)
    return false;

child.setY( (int)(child.getY() + (dVerticalScroll * mYmultiplier) ) );

return true;
}

```

[带视差效果的背景图像完整文件](#)

现在进入结尾：自定义底部弹出层行为

要实现这三个步骤，首先你需要了解默认的 BottomSheetBehavior 有五种状态：

STATE_DRAGGING、STATE_SETTLING、STATE_EXPANDED、STATE_COLLAPSED、STATE_HIDDEN，对于谷歌地图的行为，你需要在折叠和展开之间添加一个中间状态：STATE_ANCHOR_POINT。

我尝试扩展默认的 bottomSheetBehavior 但没有成功，所以我直接复制粘贴了所有代码并修改了我需要的部分。

要实现我说的功能，请按照以下步骤操作：

1. 创建一个 Java 类并继承自CoordinatorLayout.Behavior<V>
2. 将默认的BottomSheetBehavior文件中的代码复制粘贴到你的新文件中。
3. 用以下代码修改方法clampViewPositionVertical：

```

@Override
public int clampViewPositionVertical(View child, int top, int dy) {
    return constrain(top, mMinOffset, mHideable ? mParentHeight : mMaxOffset);
}
int constrain(int amount, int low, int high) {
    return amount < low ? low : (amount > high ? high : amount);
}

```

4. 添加一个新状态

```
public static final int STATE_ANCHOR_POINT = X;
```

5. 修改以下方法：onLayoutChild、onStopNestedScroll、BottomSheetBehavior<V>中的(V view)以及setState（可选）

```

    return true;
}

float dVerticalScroll = dependency.getY() - mPreviousY;
mPreviousY = dependency.getY();

//going up
if (dVerticalScroll <= 0 && child.getY() <= 0) {
    child.setY(0);
    return true;
}

//going down
if (dVerticalScroll >= 0 && dependency.getY() <= mImageHeight)
    return false;

child.setY( (int)(child.getY() + (dVerticalScroll * mYmultiplier) ) );

return true;
}

```

[The complete file for backdrop image with parallax effect](#)

Now for the end: **The Custom BottomSheet Behavior**

To achieve the 3 steps, first you need to understand that default BottomSheetBehavior has 5 states:

STATE_DRAGGING, STATE_SETTLING, STATE_EXPANDED, STATE_COLLAPSED, STATE_HIDDEN and for the Google Maps behavior you need to add a middle state between collapsed and expanded: STATE_ANCHOR_POINT.

I tried to extend the default bottomSheetBehavior with no success, so I just copy pasted all the code and modified what I need.

To achieve what I'm talking about follow the next steps:

1. Create a Java class and extend it from CoordinatorLayout.Behavior<V>
2. Copy paste code from default BottomSheetBehavior file to your new one.
3. Modify the method clampViewPositionVertical with the following code:

```

@Override
public int clampViewPositionVertical(View child, int top, int dy) {
    return constrain(top, mMinOffset, mHideable ? mParentHeight : mMaxOffset);
}
int constrain(int amount, int low, int high) {
    return amount < low ? low : (amount > high ? high : amount);
}

```

4. Add a new state

```
public static final int STATE_ANCHOR_POINT = X;
```

5. Modify the next methods: onLayoutChild, onStopNestedScroll, BottomSheetBehavior<V> from(V view) and setState (optional)

```

public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // 先让父布局进行布局
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
            !ViewCompat.getFitsSystemWindows(child)) {
            ViewCompat.setFitsSystemWindows(child, true);
        }
    }
    parent.onLayoutChild(child, layoutDirection);
}
// 偏移底部弹出层
mParentHeight = parent.getHeight();
mMinOffset = Math.max(0, mParentHeight - child.getHeight());
mMaxOffset = Math.max(mParentHeight - mPeekHeight, mMinOffset);

//if (mState == STATE_EXPANDED) {
//    ViewCompat.offsetTopAndBottom(child, mMinOffset);
//} else if (mHideable && mState == STATE_HIDDEN...) {
if (mState == STATE_ANCHOR_POINT) {
    ViewCompat.offsetTopAndBottom(child, mAnchorPoint);
} else if (mState == STATE_EXPANDED) {
    ViewCompat.offsetTopAndBottom(child, mMinOffset);
} else if (mHideable && mState == STATE_HIDDEN) {
    ViewCompat.offsetTopAndBottom(child, mParentHeight);
} else if (mState == STATE_COLLAPSED) {
    ViewCompat.offsetTopAndBottom(child, mMaxOffset);
}
if (mViewDragHelper == null) {
    mViewDragHelper = ViewDragHelper.create(parent, mDragCallback);
}
mViewRef = new WeakReference<>(child);
mNestedScrollingChildRef = new WeakReference<>(findScrollingChild(child));
return true;
}

public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, V child, View target) {
    if (child.getTop() == mMinOffset) {
        setStateInternal(STATE_EXPANDED);
        return;
    }
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int top;
    int targetState;
    if (mLastNestedScrollDy > 0) {
        //top = mMinOffset;
        //targetState = STATE_EXPANDED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        } else {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        }
    } else if (mHideable && shouldHide(child, getYVelocity())) {
        top = mParentHeight;
        targetState = STATE_HIDDEN;
    } else if (mLastNestedScrollDy == 0) {
        int currentTop = child.getTop();

```

```

public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // First let the parent lay it out
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
            !ViewCompat.getFitsSystemWindows(child)) {
            ViewCompat.setFitsSystemWindows(child, true);
        }
    }
    parent.onLayoutChild(child, layoutDirection);
}
// Offset the bottom sheet
mParentHeight = parent.getHeight();
mMinOffset = Math.max(0, mParentHeight - child.getHeight());
mMaxOffset = Math.max(mParentHeight - mPeekHeight, mMinOffset);

//if (mState == STATE_EXPANDED) {
//    ViewCompat.offsetTopAndBottom(child, mMinOffset);
//} else if (mHideable && mState == STATE_HIDDEN...) {
if (mState == STATE_ANCHOR_POINT) {
    ViewCompat.offsetTopAndBottom(child, mAnchorPoint);
} else if (mState == STATE_EXPANDED) {
    ViewCompat.offsetTopAndBottom(child, mMinOffset);
} else if (mHideable && mState == STATE_HIDDEN) {
    ViewCompat.offsetTopAndBottom(child, mParentHeight);
} else if (mState == STATE_COLLAPSED) {
    ViewCompat.offsetTopAndBottom(child, mMaxOffset);
}
if (mViewDragHelper == null) {
    mViewDragHelper = ViewDragHelper.create(parent, mDragCallback);
}
mViewRef = new WeakReference<>(child);
mNestedScrollingChildRef = new WeakReference<>(findScrollingChild(child));
return true;
}

public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, V child, View target) {
    if (child.getTop() == mMinOffset) {
        setStateInternal(STATE_EXPANDED);
        return;
    }
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int top;
    int targetState;
    if (mLastNestedScrollDy > 0) {
        //top = mMinOffset;
        //targetState = STATE_EXPANDED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        } else {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        }
    } else if (mHideable && shouldHide(child, getYVelocity())) {
        top = mParentHeight;
        targetState = STATE_HIDDEN;
    } else if (mLastNestedScrollDy == 0) {
        int currentTop = child.getTop();

```

```

        if (Math.abs(currentTop - mMinOffset) < Math.abs(currentTop - mMaxOffset)) {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        } else {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
    } else {
        //top = mMaxOffset;
        //targetState = STATE_COLLAPSED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        } else {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
    }
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        setStateInternal(STATE_SETTLING);
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, targetState));
    } else {
        setStateInternal(targetState);
    }
    mNestedScrolled = false;
}

public final void setState(@State int state) {
    if (state == mState) {
        return;
    }
    if (mViewRef == null) {
        // 视图尚未布局；修改 mState 并让 onLayoutChild 稍后处理
        /**
         * 新行为 (新增: state == STATE_ANCHOR_POINT)
         */
        if (state == STATE_COLLAPSED || state == STATE_EXPANDED ||
            state == STATE_ANCHOR_POINT ||
            (mHideable && state == STATE_HIDDEN)) {
            mState = state;
        }
        return;
    }
    V child = mViewRef.get();
    if (child == null) {
        return;
    }
    int top;
    if (state == STATE_COLLAPSED) {
        top = mMaxOffset;
    } else if (state == STATE_ANCHOR_POINT) {
        top = mAnchorPoint;
    } else if (state == STATE_EXPANDED) {
        top = mMinOffset;
    } else if (mHideable && state == STATE_HIDDEN) {
        top = mParentHeight;
    } else {
        throw new IllegalArgumentException("Illegal state argument: " + state);
    }
    setStateInternal(STATE_SETTLING);
}

```

```

        if (Math.abs(currentTop - mMinOffset) < Math.abs(currentTop - mMaxOffset)) {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        } else {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
    } else {
        //top = mMaxOffset;
        //targetState = STATE_COLLAPSED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        } else {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
    }
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        setStateInternal(STATE_SETTLING);
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, targetState));
    } else {
        setStateInternal(targetState);
    }
    mNestedScrolled = false;
}

public final void setState(@State int state) {
    if (state == mState) {
        return;
    }
    if (mViewRef == null) {
        // The view is not laid out yet; modify mState and let onLayoutChild handle it later
        /**
         * New behavior (added: state == STATE_ANCHOR_POINT)
         */
        if (state == STATE_COLLAPSED || state == STATE_EXPANDED ||
            state == STATE_ANCHOR_POINT ||
            (mHideable && state == STATE_HIDDEN)) {
            mState = state;
        }
        return;
    }
    V child = mViewRef.get();
    if (child == null) {
        return;
    }
    int top;
    if (state == STATE_COLLAPSED) {
        top = mMaxOffset;
    } else if (state == STATE_ANCHOR_POINT) {
        top = mAnchorPoint;
    } else if (state == STATE_EXPANDED) {
        top = mMinOffset;
    } else if (mHideable && state == STATE_HIDDEN) {
        top = mParentHeight;
    } else {
        throw new IllegalArgumentException("Illegal state argument: " + state);
    }
    setStateInternal(STATE_SETTLING);
}

```

```

if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
    ViewCompat.postOnAnimation(child, new SettleRunnable(child, state));
}
}

public static <V extends View> BottomSheetBehaviorGoogleMapsLike<V> from(V view) {
    ViewGroup.LayoutParams params = view.getLayoutParams();
    if (!(params instanceof CoordinatorLayout.LayoutParams)) {
        throw new IllegalArgumentException("该视图不是 CoordinatorLayout 的子视图");
    }
    CoordinatorLayout.Behavior behavior = ((CoordinatorLayout.LayoutParams) params)
        .getBehavior();
    if (!(behavior instanceof BottomSheetBehaviorGoogleMapsLike)) {
        throw new IllegalArgumentException(
            "该视图未关联 BottomSheetBehaviorGoogleMapsLike");
    }
    return (BottomSheetBehaviorGoogleMapsLike<V>) behavior;
}

```

[查看包含所有自定义行为的完整项目链接](#)

它的外观如下：



```

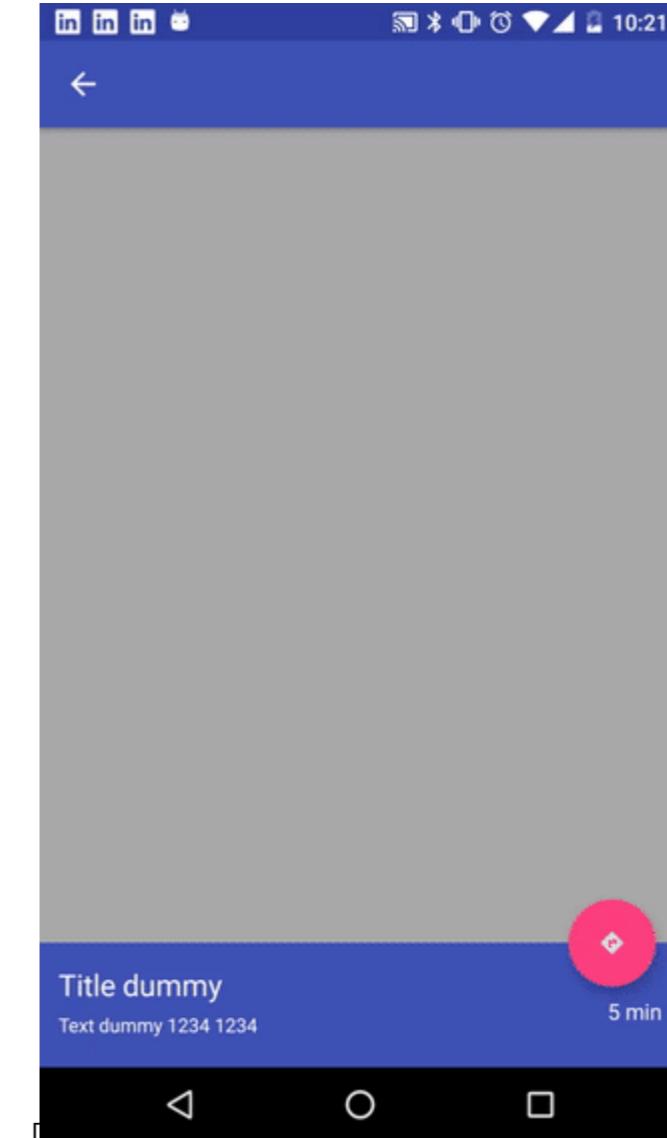
if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
    ViewCompat.postOnAnimation(child, new SettleRunnable(child, state));
}
}

public static <V extends View> BottomSheetBehaviorGoogleMapsLike<V> from(V view) {
    ViewGroup.LayoutParams params = view.getLayoutParams();
    if (!(params instanceof CoordinatorLayout.LayoutParams)) {
        throw new IllegalArgumentException("The view is not a child of CoordinatorLayout");
    }
    CoordinatorLayout.Behavior behavior = ((CoordinatorLayout.LayoutParams) params)
        .getBehavior();
    if (!(behavior instanceof BottomSheetBehaviorGoogleMapsLike)) {
        throw new IllegalArgumentException(
            "The view is not associated with BottomSheetBehaviorGoogleMapsLike");
    }
    return (BottomSheetBehaviorGoogleMapsLike<V>) behavior;
}

```

[Link to the whole project](#) where you can see all the Custom Behaviors

And here it is how it looks like:



第153.3节：使用 BottomSheetDialog 的模态底部弹窗

BottomSheetDialog 是一种样式为底部弹窗的对话框

只需使用：

```
//创建一个新的 BottomSheetDialog
BottomSheetDialog 对话框 = new BottomSheetDialog(context);
//加载布局 R.layout.my_dialog_layout
dialog.setContentView(R.layout.my_dialog_layout);
//显示对话框
dialog.show();
```

在这种情况下，你不需要附加 BottomSheet 行为。

第153.4节：使用 BottomSheetDialogFragment 的模态底部弹窗

你可以使用 BottomSheetDialogFragment 实现一个 模态底部弹窗。

BottomSheetDialogFragment 是一个模态底部弹窗。

这是 DialogFragment 的一个版本，使用 BottomSheetDialog 显示底部弹窗，而不是浮动对话框。

只需定义片段：

```
public class MyBottomSheetDialogFragment extends BottomSheetDialogFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        return inflater.inflate(R.layout.my_fragment_bottom_sheet, container);
    }
}
```

然后使用以下代码显示该片段：

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

该片段将创建一个BottomSheetDialog。

第153.5节：持久性底部面板

您可以通过将BottomSheetBehavior附加到CoordinatorLayout的子视图来实现持久性底部面板：

```
<android.support.design.widget.CoordinatorLayout >

<!-- . . . -->

<LinearLayout
    android:id="@+id/bottom_sheet"
    android:elevation="4dp"
    android:minHeight="120dp"
```

Section 153.3: Modal bottom sheets with BottomSheetDialog

The [BottomSheetDialog](#) is a dialog styled as a bottom sheet

Just use:

```
//Create a new BottomSheetDialog
BottomSheetDialog dialog = new BottomSheetDialog(context);
//Inflate the layout R.layout.my_dialog_layout
dialog.setContentView(R.layout.my_dialog_layout);
//Show the dialog
dialog.show();
```

In this case you don't need to attach a BottomSheet behavior.

Section 153.4: Modal bottom sheets with BottomSheetDialogFragment

You can realize a [modal bottom sheets](#) using a [BottomSheetDialogFragment](#).

The [BottomSheetDialogFragment](#) is a modal bottom sheet.

This is a version of DialogFragment that shows a bottom sheet using BottomSheetDialog instead of a floating dialog.

Just define the fragment:

```
public class MyBottomSheetDialogFragment extends BottomSheetDialogFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        return inflater.inflate(R.layout.my_fragment_bottom_sheet, container);
    }
}
```

Then use this code to show the fragment:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

This Fragment will create a BottomSheetDialog.

Section 153.5: Persistent Bottom Sheets

You can achieve a [Persistent Bottom Sheet](#) attaching a [BottomSheetBehavior](#) to a child View of a [CoordinatorLayout](#):

```
<android.support.design.widget.CoordinatorLayout >

<!-- . . . -->

<LinearLayout
    android:id="@+id/bottom_sheet"
    android:elevation="4dp"
    android:minHeight="120dp"
```

```

    app:behavior_peekHeight="120dp"
...
    app:layout_behavior="android.support.design.widget.BottomSheetBehavior">
        <!-- . . . -->
    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>

```

然后在你的代码中，你可以使用以下方式创建一个引用：

```

// 带有 BottomSheetBehavior 的视图
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);

```

您可以使用setState()方法设置BottomSheetBehavior的状态：

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

您可以使用以下状态之一：

- STATE_COLLAPSED：此折叠状态为默认状态，仅显示布局底部的一部分。
高度可以通过app:behavior_peekHeight属性控制（默认值为0）
- STATE_EXPANDED：底部面板的完全展开状态，此时整个底部面板可见
(如果其高度小于包含的CoordinatorLayout) 或填满整个CoordinatorLayout
- STATE_HIDDEN：默认禁用（可通过app:behavior_hideable属性启用），启用后
允许用户向下滑动底部面板以完全隐藏底部面板

如果您想接收状态变化的回调，可以添加一个BottomSheetCallback：

```

mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
        // 响应状态变化
    }
    @Override
    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        // 响应拖动事件
    }
});

```

第153.6节：默认以展开模式打开底部弹出框DialogFragment

底部弹出框DialogFragment默认以STATE_COLLAPSED状态打开。可以通过以下代码模板强制其以STATE_EXPANDED状态打开，并占满整个设备屏幕。

```

@NonNull @Override public Dialog onCreateDialog(Bundle savedInstanceState) {

    BottomSheetDialog dialog = (BottomSheetDialog) super.onCreateDialog(savedInstanceState);

    dialog.setOnShowListener(new DialogInterface.OnShowListener() {
        @Override
        public void onShow(DialogInterface dialog) {

```

```

    app:behavior_peekHeight="120dp"
...
    app:layout_behavior="android.support.design.widget.BottomSheetBehavior">
        <!-- . . . -->
    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>

```

Then in your code you can create a reference using:

```

// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);

```

You can set the state of your BottomSheetBehavior using the [setState\(\)](#) method:

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

You can use one of these states:

- STATE_COLLAPSED: this collapsed state is the default and shows just a portion of the layout along the bottom.
The height can be controlled with the app:behavior_peekHeight attribute (defaults to 0)
- STATE_EXPANDED: the fully expanded state of the bottom sheet, where either the whole bottom sheet is visible
(if its height is less than the containing CoordinatorLayout) or the entire CoordinatorLayout is filled
- STATE_HIDDEN: disabled by default (and enabled with the app:behavior_hideable attribute), enabling this
allows users to swipe down on the bottom sheet to completely hide the bottom sheet

If you'd like to receive callbacks of state changes, you can add a BottomSheetCallback:

```

mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
        // React to state change
    }
    @Override
    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        // React to dragging events
    }
});

```

Section 153.6: Open BottomSheet DialogFragment in Expanded mode by default

BottomSheet DialogFragment opens up in STATE_COLLAPSED by default. Which can be forced to open to STATE_EXPANDED and take up the full device screen with help of the following code template.

```

@NonNull @Override public Dialog onCreateDialog(Bundle savedInstanceState) {

    BottomSheetDialog dialog = (BottomSheetDialog) super.onCreateDialog(savedInstanceState);

    dialog.setOnShowListener(new DialogInterface.OnShowListener() {
        @Override
        public void onShow(DialogInterface dialog) {

```

```
BottomSheetDialog d = (BottomSheetDialog) dialog;

    FrameLayout bottomSheet = (FrameLayout)
d.findViewById(android.support.design.R.id.design_bottom_sheet);
    BottomSheetBehavior.from(bottomSheet).setState(BottomSheetBehavior.STATE_EXPANDED);
}
});

// 对你的对话框执行一些操作，比如设置内容视图等
return dialog;
}
```

虽然对动画略显明显，但在全屏打开DialogFragment的任务上表现非常出色。

```
BottomSheetDialog d = (BottomSheetDialog) dialog;

    FrameLayout bottomSheet = (FrameLayout)
d.findViewById(android.support.design.R.id.design_bottom_sheet);
    BottomSheetBehavior.from(bottomSheet).setState(BottomSheetBehavior.STATE_EXPANDED);
}
});

// Do something with your dialog like setContentView() or whatever
return dialog;
}
```

Although dialog animation is slightly noticeable but does the task of opening the DialogFragment in full screen very well.

第154章：EditText

第154.1节：使用EditText

EditText是Android应用中标准的文本输入控件。如果用户需要在应用中输入文本，这是他们进行输入的主要方式。

EditText

有许多重要属性可以设置以自定义EditText的行为。以下列出了一些。更多输入字段的详细信息请参阅官方文本字段指南。

用法

使用以下XML可以将一个具有所有默认行为的EditText添加到布局中：

```
<EditText  
    android:id="@+id/et_simple"  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent">  
</EditText>
```

注意，*EditText*只是*TextView*的一个轻量扩展，继承了所有相同的属性。

获取值

获取输入到 EditText 中的文本值如下：

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);  
String strValue = simpleEditText.getText().toString();
```

进一步的输入自定义

我们可能想限制输入为单行文本（避免换行）：

```
<EditText  
    android:singleLine="true"  
    android:lines="1"  
>
```

你可以使用 `digits` 属性限制字段中可输入的字符：

```
<EditText  
    android:inputType="number"  
    android:digits="01"  
>
```

这将限制输入的数字仅为“0”和“1”。我们可能还想限制总字符数，使用：

```
<EditText  
    android:maxLength="5"  
>
```

通过使用这些属性，我们可以定义文本字段的预期输入行为。

Chapter 154: EditText

Section 154.1: Working with EditTexts

The EditText is the standard text entry widget in Android apps. If the user needs to enter text into an app, this is the primary way for them to do that.

EditText

There are many important properties that can be set to customize the behavior of an EditText. Several of these are listed below. Check out the official text fields guide for even more input field details.

Usage

An EditText is added to a layout with all default behaviors with the following XML:

```
<EditText  
    android:id="@+id/et_simple"  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent">  
</EditText>
```

Note that an EditText is simply a thin extension of the TextView and inherits all of the same properties.

Retrieving the Value

Getting the value of the text entered into an EditText is as follows:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);  
String strValue = simpleEditText.getText().toString();
```

Further Entry Customization

We might want to limit the entry to a single-line of text (avoid newlines):

```
<EditText  
    android:singleLine="true"  
    android:lines="1"  
>
```

You can limit the characters that can be entered into a field using the `digits` attribute:

```
<EditText  
    android:inputType="number"  
    android:digits="01"  
>
```

This would restrict the digits entered to just "0" and "1". We might want to limit the total number of characters with:

```
<EditText  
    android:maxLength="5"  
>
```

Using these properties we can define the expected input behavior for text fields.

调整颜色

您可以通过以下属性调整 EditText 中选中文本的高亮背景颜色：

```
<EditText  
    android:textColorHighlight="#7cff88"  
/>
```

显示占位提示

您可能想为 EditText 控件设置提示，以引导用户输入特定内容，方法如下：

```
<EditText  
    ...  
    android:hint="@string/my_hint"  
/>
```

提示

更改底部线条颜色

假设您正在使用 AppCompat 库，可以覆盖样式中的 colorControlNormal、colorControlActivated 和 colorControlHighlight：

```
<style name="Theme.App.Base" parent="Theme.AppCompat.Light.DarkActionBar">  
    <item name="colorControlNormal">#d32f2f</item>  
    <item name="colorControlActivated">#ff5722</item>  
    <item name="colorControlHighlight">#f44336</item>  
</style>
```

如果您在 DialogFragment 中看不到这些样式被应用，已知在使用传入 onCreateView() 方法的 LayoutInflater 时存在一个错误。

该问题已在 AppCompat v23 库中修复。请参阅此指南了解如何升级。另一个临时解决方法是使用 Activity 的布局填充器，而不是传入 onCreateView() 方法的那个布局填充器：

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment, container);  
}
```

监听 EditText 输入

查看基本事件监听笔记，了解如何监听 EditText 的变化并在变化发生时执行操作。

显示浮动标签反馈

传统上，用户开始输入后，EditText 会隐藏提示信息（如上所述）。此外，任何验证错误信息都必须由开发者手动管理。

使用 TextInputLayout，您可以设置浮动标签来显示提示和错误信息。更多详情请见此处。

Adjusting Colors

You can adjust the highlight background color of selected text within an EditText with the android:textColorHighlight property:

```
<EditText  
    android:textColorHighlight="#7cff88"  
/>
```

Displaying Placeholder Hints

You may want to set the hint for the EditText control to prompt a user for specific input with:

```
<EditText  
    ...  
    android:hint="@string/my_hint"  
/>
```

Hints

Changing the bottom line color

Assuming you are using the AppCompat library, you can override the styles colorControlNormal, colorControlActivated, and colorControlHighlight:

```
<style name="Theme.App.Base" parent="Theme.AppCompat.Light.DarkActionBar">  
    <item name="colorControlNormal">#d32f2f</item>  
    <item name="colorControlActivated">#ff5722</item>  
    <item name="colorControlHighlight">#f44336</item>  
</style>
```

If you do not see these styles applied within a DialogFragment, there is a known bug when using the LayoutInflater passed into the onCreateView() method.

The issue has already been fixed in the AppCompat v23 library. See this guide about how to upgrade. Another temporary workaround is to use the Activity's layout inflator instead of the one passed into the onCreateView() method:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment, container);  
}
```

Listening for EditText Input

Check out the basic event listeners cliffnotes for a look at how to listen for changes to an EditText and perform an action when those changes occur.

Displaying Floating Label Feedback

Traditionally, the EditText hides the hint message (explained above) after the user starts typing. In addition, any validation error messages had to be managed manually by the developer.

With the TextInputLayout you can setup a floating label to display hints and error messages. You can find more details here.

第154.2节：自定义InputType

文本字段可以有不同的输入类型，例如数字、日期、密码或电子邮件地址。类型决定了字段中允许输入的字符种类，并可能提示虚拟键盘优化其布局以便更频繁使用的字符。

默认情况下，任何位于EditText控件内的文本内容都会以纯文本显示。通过设置属性，我们可以方便地输入不同类型的信息，如电话号码和密码：

```
<EditText  
    ...  
    android:inputType="phone"  
/>
```

最常见的输入类型包括：

类型	描述
textUri	将用作URI的文本
textEmailAddress	将用作电子邮件地址的文本
textPersonName	是人的姓名的文本
文本密码	应被遮蔽的密码文本
数字	仅限数字的字段
电话	用于输入电话号码
日期	用于输入日期
时间	用于输入时间
textMultiLine	允许字段中输入多行文本

android:inputType 还允许您指定某些键盘行为，例如是否将所有新单词首字母大写，或使用自动完成和拼写建议等功能。

以下是定义键盘行为的一些常见输入类型值：

类型	描述
textCapSentences	普通文本键盘，会将每个新句子的首字母大写
textCapWords	普通文本键盘，会将每个单词首字母大写。适用于标题或人名
textAutoCorrect	普通文本键盘，会纠正常见的拼写错误

如果需要，您可以设置多个inputType属性（用'|'分隔）。

示例：

```
<EditText  
    android:id="@+id/postal_address"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/postal_address_hint"  
    android:inputType="textPostalAddress|  
        textCapWords|  
        textNoSuggestions" />
```

您可以在此处查看所有可用输入类型的列表。[_](#)

第154.3节：自定义编辑文本中的图标或按钮及其

Section 154.2: Customizing the InputType

Text fields can have different input types, such as number, date, password, or email address. The type determines what kind of characters are allowed inside the field, and may prompt the virtual keyboard to optimize its layout for frequently used characters.

By default, any text contents within an EditText control is displayed as plain text. By setting the `inputType` attribute, we can facilitate input of different types of information, like phone numbers and passwords:

```
<EditText  
    ...  
    android:inputType="phone"  
/>
```

Most common input types include:

Type	Description
textUri	Text that will be used as a URI
textEmailAddress	Text that will be used as an e-mail address
textPersonName	Text that is the name of a person
textPassword	Text that is a password that should be obscured
number	A numeric only field
phone	For entering a phone number
date	For entering a date
time	For entering a time
textMultiLine	Allow multiple lines of text in the field

The `android:inputType` also allows you to specify certain keyboard behaviors, such as whether to capitalize all new words or use features like auto-complete and spelling suggestions.

Here are some of the common input type values that define keyboard behaviors:

Type	Description
textCapSentences	Normal text keyboard that capitalizes the first letter for each new sentence
textCapWords	Normal text keyboard that capitalizes every word. Good for titles or person names
textAutoCorrect	Normal text keyboard that corrects commonly misspelled words

You can set multiple `inputType` attributes if needed (separated by '|').

Example:

```
<EditText  
    android:id="@+id/postal_address"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/postal_address_hint"  
    android:inputType="textPostalAddress|  
        textCapWords|  
        textNoSuggestions" />
```

You can see a list of all available input types [here](#).

Section 154.3: Icon or button inside Custom Edit Text and its

动作和点击监听器

此示例将帮助实现右侧带图标的编辑文本框。

注意：这里我只使用了setCompoundDrawablesWithIntrinsicBounds，因此如果你想更改图标位置，可以在setIcon中使用setCompoundDrawablesWithIntrinsicBounds来实现。

```
public class MKEditText extends AppCompatEditText {

    public interface IconClickListener {
        public void onClick();
    }

    private IconClickListener mIconClickListener;

    private static final String TAG = MKEditText.class.getSimpleName();

    private final int EXTRA_TOUCH_AREA = 50;
    private Drawable mDrawable;
    private boolean touchDown;

    public MKEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MKEditText(Context context) {
        super(context);
    }

    public MKEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public void showRightIcon() {
        mDrawable = ContextCompat.getDrawable(getContext(), R.drawable.ic_android_black_24dp);

        setIcon();
    }

    public void setIconClickListener(IconClickListener iconClickListener) {
        mIconClickListener = iconClickListener;
    }

    private void setIcon() {
        Drawable[] drawables = getCompoundDrawables();

        setCompoundDrawablesWithIntrinsicBounds(drawables[0], drawables[1], mDrawable,
                drawables[3]);

        setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        setSelection(getText().length());
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        final int right = getRight();
        final int drawableSize = getCompoundPaddingRight();
        final int x = (int) event.getX();
        switch (event.getAction()) {
```

action and click listeners

This example will help to have the Edit text with the icon at the right side.

Note: In this just I am using setCompoundDrawablesWithIntrinsicBounds, So if you want to change the icon position you can achieve that using setCompoundDrawablesWithIntrinsicBounds in setIcon.

```
public class MKEditText extends AppCompatEditText {

    public interface IconClickListener {
        public void onClick();
    }

    private IconClickListener mIconClickListener;

    private static final String TAG = MKEditText.class.getSimpleName();

    private final int EXTRA_TOUCH_AREA = 50;
    private Drawable mDrawable;
    private boolean touchDown;

    public MKEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MKEditText(Context context) {
        super(context);
    }

    public MKEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public void showRightIcon() {
        mDrawable = ContextCompat.getDrawable(getContext(), R.drawable.ic_android_black_24dp);

        setIcon();
    }

    public void setIconClickListener(IconClickListener iconClickListener) {
        mIconClickListener = iconClickListener;
    }

    private void setIcon() {
        Drawable[] drawables = getCompoundDrawables();

        setCompoundDrawablesWithIntrinsicBounds(drawables[0], drawables[1], mDrawable,
                drawables[3]);

        setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        setSelection(getText().length());
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        final int right = getRight();
        final int drawableSize = getCompoundPaddingRight();
        final int x = (int) event.getX();
        switch (event.getAction()) {
```

```

case MotionEvent.ACTION_DOWN:
    if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA)
{
    touchDown = true;
        return true;
    }
    break;
case MotionEvent.ACTION_UP:
    if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA
&& touchDown) {
touchDown = false;
        if (mIconClickListener != null) {
            mIconClickListener.onClick();
        }
        return true;
    }
    touchDown = false;
    break;

}
return super.onTouchEvent(event);
}

```

如果您想更改触摸区域，可以更改我默认设置为50的EXTRA_TOUCH_AREA值。

要启用按钮和点击监听器，可以从您的Activity或Fragment中这样调用，

```

MKEditText mkEditText = (MKEditText) findViewById(R.id.password);
mkEditText.showRightIcon();
mkEditText.setOnClickListener(new MKEditText.IconClickListener() {
    @Override
    public void onClick() {
        // 您可以在这里为图标执行操作。
    }
});

```

第154.4节：隐藏软键盘

隐藏软键盘是使用EditText时的一个基本需求。软键盘默认情况下只能通过按返回键关闭，因此大多数开发者使用InputMethodManager来强制Android隐藏虚拟键盘，调用hideSoftInputFromWindow并传入包含当前焦点视图的窗口令牌。实现如下代码：

```

public void hideSoftKeyboard()
{
InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}

```

代码很直接，但另一个主要问题是隐藏函数需要在某个事件发生时调用。当您需要在点击EditText以外的任何地方时隐藏软键盘，该怎么办？以下代码提供了一个简洁的函数，只需在您的onCreate()方法中调用一次即可。

```

case MotionEvent.ACTION_DOWN:
    if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA)
{
    touchDown = true;
        return true;
    }
    break;
case MotionEvent.ACTION_UP:
    if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA
&& touchDown) {
        touchDown = false;
        if (mIconClickListener != null) {
            mIconClickListener.onClick();
        }
        return true;
    }
    touchDown = false;
    break;

}
return super.onTouchEvent(event);
}

```

If you want to change the touch area you can change the EXTRA_TOUCH_AREA values default I gave as 50.

And for Enable the button and click listener you can call from your Activity or Fragment like this,

```

MKEditText mkEditText = (MKEditText) findViewById(R.id.password);
mkEditText.showRightIcon();
mkEditText.setOnClickListener(new MKEditText.IconClickListener() {
    @Override
    public void onClick() {
        // You can do action here for the icon.
    }
});

```

Section 154.4: Hiding SoftKeyboard

Hiding Softkeyboard is a **basic requirement** usually when working with EditText. The softkeyboard by *default* can only be closed by pressing back button and so most developers use [InputMethodManager](#) to force Android to hide the virtual keyboard calling [hideSoftInputFromWindow](#) and passing in the token of the window containing your focused view. The code to do the following:

```

public void hideSoftKeyboard()
{
    InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
    inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}

```

The code is direct, but another major problems that arises is that the hide function needs to be called when some event occurs. What to do when you need the Softkeyboard hidden upon pressing anywhere other than your EditText? The following code gives a neat function that needs to be called in your onCreate() method just once.

```

public void setupUI(View view)
{
    String s = "inside";
    //为非文本框视图设置触摸监听器以隐藏键盘。
    if (!(view instanceof EditText)) {

        view.setOnTouchListener(new View.OnTouchListener() {

            public boolean onTouch(View v, MotionEvent event) {
                hideSoftKeyboard();
                return false;
            }
        });
    }

    //如果是布局容器，遍历子视图并递归处理。
    if (view instanceof ViewGroup) {

        for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {

            View innerView = ((ViewGroup) view).getChildAt(i);

            setupUI(innerView);
        }
    }
}

```

```

public void setupUI(View view)
{
    String s = "inside";
    //Set up touch listener for non-text box views to hide keyboard.
    if (!(view instanceof EditText)) {

        view.setOnTouchListener(new View.OnTouchListener() {

            public boolean onTouch(View v, MotionEvent event) {
                hideSoftKeyboard();
                return false;
            }
        });
    }

    //If a layout container, iterate over children and seed recursion.
    if (view instanceof ViewGroup) {

        for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {

            View innerView = ((ViewGroup) view).getChildAt(i);

            setupUI(innerView);
        }
    }
}

```

第154.5节：`inputType` 属性

EditText控件中的inputType属性：(在Android 4.4.3和2.3.3上测试)

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage= 键盘：字母/默认。回车键：发送/下一步。表情：是。大小写：小写。
建议：是。附加字符：， 和 . 以及所有字符

textFilter= 键盘：字母/默认。回车键：发送/下一步。表情：是。大小写：小写。建议：否。
附加字符：， 和 . 以及所有字符

textCapWords= 键盘：字母/默认。回车键：发送/下一步。表情：是。大小写：驼峰式。
建议：是。附加字符：， 和 . 以及所有字符

textCapSentences= 键盘：字母/默认。回车键：发送/下一步。表情：是。大小写：句首大写。
建议：是。附加字符：， 和 . 以及所有字符

time= 键盘：数字键。回车键：发送/下一步。表情：无。大小写：无。建议：无。添加字符：：

textMultiLine= 键盘：字母/默认。回车键：换行。表情：有。大小写：小写。建议：
是。附加字符：， 和 . 以及所有字符

number= 键盘：数字键。回车键：发送/下一步。表情：无。大小写：无。建议：无。添加字符：无

textEmailAddress= 键盘：字母/默认。回车键：发送/下一步。表情：无。大小写：小写。
建议：无。附加字符：@ 和 . 以及所有字符

(无类型)= 键盘：字母/默认。回车键：换行。表情：有。大小写：小写。建议：有。

Section 154.5: `inputType` attribute

inputType attribute in EditText widget: (tested on Android 4.4.3 and 2.3.3)

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase.
Suggestion: yes. Add. chars: , and . and everything

textFilter= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. **Suggestion: no.**
Add. chars: , and . and everything

textCapWords= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: Camel Case.**
Suggestion: yes. Add. chars: , and . and everything

textCapSentences= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: Sentence case.**
Suggestion: yes. Add. chars: , and . and everything

time= Keyboard: numeric. Enter button: Send/Next. Emotion: no. Case: -. **Suggestion: no.** Add. chars: :

textMultiLine= Keyboard: alphabet/default. **Enter button: nextline.** Emotion: yes. Case: lowercase. Suggestion:
yes. Add. chars: , and . and everything

number= **Keyboard: numeric.** Enter button: Send/Next. Emotion: no. Case: -. Suggestion: no. **Add. chars: nothing**

textEmailAddress= Keyboard: alphabet/default. Enter button: Send/Next. **Emotion: no.** Case: lowercase.
Suggestion: no. Add. chars: @ and . and everything

(No type)= Keyboard: alphabet/default. **Enter button: nextline.** Emotion: yes. Case: lowercase. Suggestion: yes.

附加字符：, 和 . 以及所有字符

textPassword= 键盘：字母/默认。回车键：发送/下一步。表情：无。大小写：小写。建议：
无。添加字符：, 和 . 以及所有字符

text= 键盘：字母/默认。回车键：发送/下一步。表情：有。大小写：小写。建议：
是。添加字符：, 和 . 以及所有字符

textShortMessage= 键盘：字母/默认。回车键：表情。表情：是。大小写：小写。
建议：是。附加字符：, 和 . 以及所有字符

textUri= 键盘：字母/默认。回车键：发送/下一步。表情：无。大小写：小写。建议：无。添加。
字符：/ 和 . 以及所有内容

textCapCharacters= 键盘：字母/默认。回车键：发送/下一步。表情：有。大小写：大写。
建议：是。附加字符：, 和 . 以及所有字符

phone= 键盘：数字。回车键：发送/下一步。表情：无。大小写：-。建议：无。添加。字符：*** # . - /
() W P N , +**

textPersonName= 键盘：字母/默认。回车键：发送/下一步。表情：有。大小写：小写。
建议：是。附加字符：, 和 . 以及所有字符

注意：自动-大写设置将改变默认行为。

注意2：在数字键盘中，所有数字均为英文1234567890。

注意3：纠正/建议设置将改变默认行为。

Add. chars: , and . and everything

textPassword= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: no. Case: lowercase. **Suggestion: no.** Add. chars: , and . and everything

text= Keyboard: Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: , and . and everything

textShortMessage= Keyboard: alphabet/default. **Enter button: emotion.** Emotion: yes. Case: lowercase.
Suggestion: yes. Add. chars: , and . and everything

textUri= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: no. Case: lowercase. **Suggestion: no.** Add.
chars: / and . and everything

textCapCharacters= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: UPPERCASE.**
Suggestion: yes. Add. chars: , and . and everything

phone= **Keyboard: numeric.** Enter button: Send/Next. Emotion: no. Case: -. **Suggestion: no.** Add. chars: *** # . - /
() W P N , +**

textPersonName= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase.
Suggestion: yes. Add. chars: , and . and everything

Note: Auto-capitalization setting will change the default behavior.

Note 2: In the Numeric keyboard, ALL numbers are English 1234567890.

Note 3: Correction/Suggestion setting will change the default behavior.

第155章：语音转文本

第155.1节：使用默认Google提示对话框的语音转文本

触发语音转文本翻译

```
private void startListening() {  
  
    // 监听用户语音输入并在同一活动中返回结果的意图  
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);  
  
    // 使用基于自由形式语音识别的语言模型。  
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,  
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);  
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());  
  
    // 对话框中显示的信息  
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,  
        getString(R.string.speech_to_text_info));  
    try {  
        startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);  
    } catch (ActivityNotFoundException a) {  
        Toast.makeText(getApplicationContext(),  
            getString(R.string.speech_not_supported),  
            Toast.LENGTH_SHORT).show();  
    }  
}
```

在 onActivityResult 中获取翻译结果

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    switch (requestCode) {  
        case REQ_CODE_SPEECH_INPUT: {  
            if (resultCode == RESULT_OK && null != data) {  
  
                ArrayList<String> result = data  
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);  
                txtSpeechInput.setText(result.get(0));  
            }  
            break;  
        }  
    }  
}
```

输出

Chapter 155: Speech to Text Conversion

Section 155.1: Speech to Text With Default Google Prompt Dialog

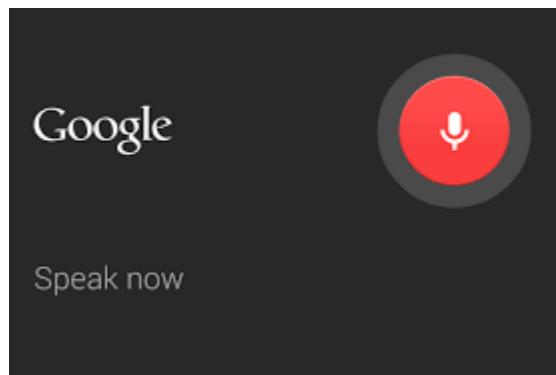
Trigger speech to text translation

```
private void startListening() {  
  
    // Intent to listen to user vocal input and return result in same activity  
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);  
  
    // Use a language model based on free-form speech recognition.  
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,  
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);  
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());  
  
    // Message to display in dialog box  
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,  
        getString(R.string.speech_to_text_info));  
    try {  
        startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);  
    } catch (ActivityNotFoundException a) {  
        Toast.makeText(getApplicationContext(),  
            getString(R.string.speech_not_supported),  
            Toast.LENGTH_SHORT).show();  
    }  
}
```

Get translated results in onActivityResult

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    switch (requestCode) {  
        case REQ_CODE_SPEECH_INPUT: {  
            if (resultCode == RESULT_OK && null != data) {  
  
                ArrayList<String> result = data  
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);  
                txtSpeechInput.setText(result.get(0));  
            }  
            break;  
        }  
    }  
}
```

Output



第155.2节：无对话框的语音转文本

以下代码可用于触发语音转文本功能，而不显示对话框：

```
public void startListeningWithoutDialog() {
    // 用于监听用户语音输入并将结果返回到同一活动的意图。
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // 使用基于自由形式语音识别的语言模型。
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
        appContext.getPackageName());

    // 添加自定义监听器。
    CustomRecognitionListener listener = new CustomRecognitionListener();
    SpeechRecognizer sr = SpeechRecognizer.createSpeechRecognizer(appContext);
    sr.setRecognitionListener(listener);
    sr.startListening(intent);
}
```

上述代码中使用的自定义监听器类CustomRecognitionListener的实现如下：

```
class CustomRecognitionListener implements RecognitionListener {
    private static final String TAG = "RecognitionListener";

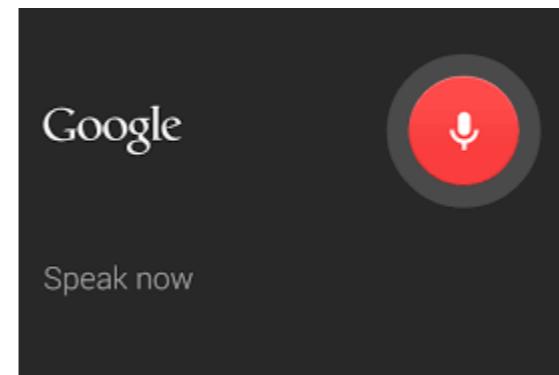
    public void onReadyForSpeech(Bundle params) {
        Log.d(TAG, "onReadyForSpeech");
    }

    public void onBeginningOfSpeech() {
        Log.d(TAG, "onBeginningOfSpeech");
    }

    public void onRmsChanged(float rmsdB) {
        Log.d(TAG, "onRmsChanged");
    }

    public void onBufferReceived(byte[] buffer) {
        Log.d(TAG, "onBufferReceived");
    }

    public void onEndOfSpeech() {
        Log.d(TAG, "onEndofSpeech");
    }
}
```



Section 155.2: Speech to Text without Dialog

The following code can be used to trigger speech-to-text translation without showing a dialog:

```
public void startListeningWithoutDialog() {
    // Intent to listen to user vocal input and return the result to the same activity.
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
        appContext.getPackageName());

    // Add custom listeners.
    CustomRecognitionListener listener = new CustomRecognitionListener();
    SpeechRecognizer sr = SpeechRecognizer.createSpeechRecognizer(appContext);
    sr.setRecognitionListener(listener);
    sr.startListening(intent);
}
```

The custom listener class CustomRecognitionListener used in the code above is implemented as follows:

```
class CustomRecognitionListener implements RecognitionListener {
    private static final String TAG = "RecognitionListener";

    public void onReadyForSpeech(Bundle params) {
        Log.d(TAG, "onReadyForSpeech");
    }

    public void onBeginningOfSpeech() {
        Log.d(TAG, "onBeginningOfSpeech");
    }

    public void onRmsChanged(float rmsdB) {
        Log.d(TAG, "onRmsChanged");
    }

    public void onBufferReceived(byte[] buffer) {
        Log.d(TAG, "onBufferReceived");
    }

    public void onEndOfSpeech() {
        Log.d(TAG, "onEndofSpeech");
    }
}
```

```
    public void onError(int error) {
        Log.e(TAG, "error " + error);

        conversionCallback.onErrorOccured(TranslatorUtil.getErrorText(error));
    }

    public void onResults(Bundle results) {
        ArrayList<String> result = data
            .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        txtSpeechInput.setText(result.get(0));
    }

    public void onPartialResults(Bundle partialResults) {
        Log.d(TAG, "onPartialResults");
    }

    public void onEvent(int eventType, Bundle params) {
        Log.d(TAG, "onEvent " + eventType);
    }
}
```

```
    public void onError(int error) {
        Log.e(TAG, "error " + error);

        conversionCallback.onErrorOccured(TranslatorUtil.getErrorText(error));
    }

    public void onResults(Bundle results) {
        ArrayList<String> result = data
            .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        txtSpeechInput.setText(result.get(0));
    }

    public void onPartialResults(Bundle partialResults) {
        Log.d(TAG, "onPartialResults");
    }

    public void onEvent(int eventType, Bundle params) {
        Log.d(TAG, "onEvent " + eventType);
    }
}
```

第156章：使用ADB安装应用程序

第156.1节：卸载应用程序

在终端中输入以下命令以卸载指定包名的应用程序：

```
adb uninstall <packagename>
```

第156.2节：安装目录中的所有apk文件

Windows :

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux :

```
for f in *.apk ; do adb install "$f" ; done
```

第156.3节：安装应用程序

在终端中输入以下命令：

```
adb install [-rtsdg] <file>
```

请注意，您必须传递一份存储在您电脑上的文件，而不是存储在您的设备上的文件。

如果你在末尾添加-r，则任何现有的冲突apk将被覆盖。否则，命令将以错误退出。

-g 将立即授予所有运行时权限。

-d 允许版本代码降级（仅适用于可调试的包）。

使用-s将应用安装到外部SD卡上。

-t 将允许使用测试应用程序。

Chapter 156: Installing apps with ADB

Section 156.1: Uninstall an app

Write the following command in your terminal to uninstall an app with a provided package name:

```
adb uninstall <packagename>
```

Section 156.2: Install all apk file in directory

Windows :

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux :

```
for f in *.apk ; do adb install "$f" ; done
```

Section 156.3: Install an app

Write the following command in your terminal:

```
adb install [-rtsdg] <file>
```

Note that you have to pass a file that is on your computer and not on your device.

If you append -r at the end, then any existing conflicting apks will be overwritten. Otherwise, the command will quit with an error.

-g will immediately grant all runtime permissions.

-d allows version code downgrade (only applicable on debuggable packages).

Use -s to install the application on the external SD card.

-t will allow to use test applications.

第157章：倒计时定时器

参数	详细信息
long millisInFuture	计时器将运行的总时长，即你希望计时器结束的未来时间。单位为毫秒。
long countDownInterval	你希望接收计时器更新的间隔。单位为毫秒。
long millisUntilFinished	在 <code>onTick()</code> 中提供的一个参数，表示倒计时计时器剩余的时间。单位为毫秒

第157.1节：创建一个简单的倒计时计时器

CountDownTimer 适用于在设定的持续时间内以固定间隔重复执行某个操作。在本例中，我们将每秒更新一次文本视图，显示剩余时间共30秒。然后当计时器结束时，我们将文本视图设置为“完成”。

```
TextView textView = (TextView) findViewById(R.id.text_view);

CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished) {
        textView.setText(String.format(Locale.getDefault(), "%d 秒.", millisUntilFinished / 1000L));
    }

    public void onFinish() {
        textView.setText("完成.");
    }
}.start();
```

第157.2节：一个更复杂的示例

在本例中，我们将根据Activity的生命周期暂停/恢复CountDownTimer。

```
private static final long TIMER_DURATION = 60000L;
private static final long TIMER_INTERVAL = 1000L;

private CountDownTimer mCountDownTimer;
private TextView textView;

private long mTimeRemaining;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textView = (TextView) findViewById(R.id.text_view); // 在xml布局中定义。

    mCountDownTimer = new CountDownTimer(TIMER_DURATION, TIMER_INTERVAL) {

        @Override
        public void onTick(long millisUntilFinished) {
            textView.setText(String.format(Locale.getDefault(), "%d秒", millisUntilFinished / 1000L));
            mTimeRemaining = millisUntilFinished; // 在Activity中保存剩余时间，用于暂停/恢复CountDownTimer。
        }
    };
}
```

Chapter 157: Count Down Timer

Parameter	Details
long millisInFuture	The total duration the timer will run for, a.k.a how far in the future you want the timer to end. In milliseconds.
long countDownInterval	The interval at which you would like to receive timer updates. In milliseconds.
long millisUntilFinished	A parameter provided in <code>onTick()</code> that tells how long the CountDownTimer has remaining. In milliseconds

Section 157.1: Creating a simple countdown timer

CountDownTimer is useful for repeatedly performing an action in a steady interval for a set duration. In this example, we will update a text view every second for 30 seconds telling how much time is remaining. Then when the timer finishes, we will set the TextView to say "Done."

```
TextView textView = (TextView) findViewById(R.id.text_view);

CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished) {
        textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished / 1000L));
    }

    public void onFinish() {
        textView.setText("Done.");
    }
}.start();
```

Section 157.2: A More Complex Example

In this example, we will pause/resume the CountDownTimer based off of the Activity lifecycle.

```
private static final long TIMER_DURATION = 60000L;
private static final long TIMER_INTERVAL = 1000L;

private CountDownTimer mCountDownTimer;
private TextView textView;

private long mTimeRemaining;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textView = (TextView) findViewById(R.id.text_view); // Define in xml layout.

    mCountDownTimer = new CountDownTimer(TIMER_DURATION, TIMER_INTERVAL) {

        @Override
        public void onTick(long millisUntilFinished) {
            textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished / 1000L));
            mTimeRemaining = millisUntilFinished; // Saving timeRemaining in Activity for pause/resume of CountDownTimer.
        }
    };
}
```

```

@Override
public void onFinish() {
    textView.setText("完成。");
}
}.start();
}

@Override
protected void onResume() {
    super.onResume();

    if (mCountDownTimer == null) { // 计时器已暂停，使用保存的时间重新创建。
        mCountDownTimer = new CountDownTimer(timeRemaining, INTERVAL) {
            @Override
            public void onTick(long millisUntilFinished) {
                textView.setText(String.format(Locale.getDefault(), "%d 秒.", millisUntilFinished / 1000L));
                timeRemaining = millisUntilFinished;
            }
        }.start();
    }

    @Override
    public void onFinish() {
        textView.setText("完成。 ");
    }
}.start();
}

@Override
protected void onPause() {
    super.onPause();
    mCountDownTimer.cancel();
    mCountDownTimer = null;
}

```

```

@Override
public void onFinish() {
    textView.setText("Done。");
}
}.start();
}

@Override
protected void onResume() {
    super.onResume();

    if (mCountDownTimer == null) { // Timer was paused, re-create with saved time.
        mCountDownTimer = new CountDownTimer(timeRemaining, INTERVAL) {
            @Override
            public void onTick(long millisUntilFinished) {
                textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished / 1000L));
                timeRemaining = millisUntilFinished;
            }
        }.start();
    }

    @Override
    public void onFinish() {
        textView.setText("Done。");
    }
}.start();
}

@Override
protected void onPause() {
    super.onPause();
    mCountDownTimer.cancel();
    mCountDownTimer = null;
}

```

第158章：条形码和二维码 读取

第158.1节：使用QRCodeReaderView（基于Zxing）

QRCodeReaderView 实现了一个Android视图，显示摄像头并在预览中检测到二维码时通知。

它使用了zxing开源的、多格式的1D/2D条码图像处理库。

将库添加到您的项目中

在build.gradle中添加QRCodeReaderView依赖

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

首次使用

- 在你的布局中添加一个QRCodeReaderView

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
    android:id="@+id/qrdecoderview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- 创建一个实现了onQRCodeReadListener的Activity，并将其用作

QrCodeReaderView的监听器。

- 确保你拥有使用该库所需的摄像头权限。

(<https://developer.android.com/training/permissions/requesting.html>)

然后在你的Activity中，可以按如下方式使用：

```
public class DecoderActivity extends Activity implements OnQRCodeReadListener {

    private TextView resultTextView;
    private QRCodeReaderView qrCodeReaderView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_decoder);

        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // 使用此函数启用/禁用解码
        qrCodeReaderView.setQRDecodingEnabled(true);

        // 使用此函数更改自动对焦间隔（默认是5秒）
        qrCodeReaderView.setAutoFocusInterval(2000L);

        // 使用此函数启用/禁用手电筒
        qrCodeReaderView.setTorchEnabled(true);

        // 使用此函数设置前置摄像头预览
        qrCodeReaderView.setFrontCamera();
    }
}
```

Chapter 158: Barcode and QR code reading

Section 158.1: Using QRCodeReaderView (based on Zxing)

QRCodeReaderView implements an Android view which show camera and notify when there's a QR code inside the preview.

It uses the [zxing](#) open-source, multi-format 1D/2D barcode image processing library.

Adding the library to your project

Add QRCodeReaderView dependency to your build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

First use

- Add to your layout a QRCodeReaderView

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
    android:id="@+id/qrdecoderview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Create an Activity which implements onQRCodeReadListener, and use it as a listener of the QRCodeReaderView.
- Make sure you have camera permissions in order to use the library.
(<https://developer.android.com/training/permissions/requesting.html>)

Then in your Activity, you can use it as follows:

```
public class DecoderActivity extends Activity implements OnQRCodeReadListener {

    private TextView resultTextView;
    private QRCodeReaderView qrCodeReaderView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_decoder);

        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // Use this function to enable/disable decoding
        qrCodeReaderView.setQRDecodingEnabled(true);

        // Use this function to change the autofocus interval (default is 5 secs)
        qrCodeReaderView.setAutoFocusInterval(2000L);

        // Use this function to enable/disable Torch
        qrCodeReaderView.setTorchEnabled(true);

        // Use this function to set front camera preview
        qrCodeReaderView.setFrontCamera();
    }
}
```

```
// 使用此函数设置后置摄像头预览
qrCodeReaderView.setBackCamera();
}

// 当二维码被解码时调用
// "text": 二维码中编码的文本
// "points": 二维码控制点在视图中的位置点
@Override
public void onQRCodeRead(String text, PointF[] points) {
    resultTextView.setText(text);
}

@Override
protected void onResume() {
    super.onResume();
qrCodeReaderView.startCamera();
}

@Override
protected void onPause() {
    super.onPause();
qrCodeReaderView.stopCamera();
}
}
```

```
// Use this function to set back camera preview
qrCodeReaderView.setBackCamera();
}

// Called when a QR is decoded
// "text" : the text encoded in QR
// "points" : points where QR control points are placed in View
@Override
public void onQRCodeRead(String text, PointF[] points) {
    resultTextView.setText(text);
}

@Override
protected void onResume() {
    super.onResume();
    qrCodeReaderView.startCamera();
}

@Override
protected void onPause() {
    super.onPause();
    qrCodeReaderView.stopCamera();
}
}
```

第159章：Android PayPal网关集成

第159.1节：在您的安卓代码中设置PayPal

1) 首先访问Paypal开发者网站并创建一个应用程序。

2) 现在打开你的清单文件并添加以下权限

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3) 以及一些必需的Activity和服务

```
<service
    android:name="compaypal.android.sdk.payments.PayPalService"
    android:exported="false" />
<activity android:name="compaypal.android.sdk.payments.PaymentActivity" />
<activity android:name="compaypal.android.sdk.payments.LoginActivity" />
<activity android:name="compaypal.android.sdk.payments.PaymentMethodActivity" />
<activity android:name="compaypal.android.sdk.payments.PaymentConfirmActivity" />
<activity android:name="compaypal.android.sdk.payments.PayPalFuturePaymentActivity" />
<activity android:name="compaypal.android.sdk.payments.FuturePaymentConsentActivity" />
<activity android:name="compaypal.android.sdk.payments.FuturePaymentInfoActivity" />
<activity
    android:name="io.card.payment.CardIOActivity"
    android:configChanges="keyboardHidden|orientation" />
<activity android:name="io.card.payment.DataEntryActivity" />
```

4) 打开你的Activity类并为你的应用设置配置

```
// 设置生产环境/沙盒环境/无网络环境
private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) 现在从 PayPal 开发者账户设置客户端 ID

```
private static final String CONFIG_CLIENT_ID = "PUT YOUR CLIENT ID";
```

6) 在 onCreate 方法内调用 PayPal 服务

```
Intent intent = new Intent(this, PayPalService.class);
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
startService(intent);
```

7) 现在你可以准备付款了，只需在按钮按下时调用支付活动

```
PayPalPayment thingToBuy = new PayPalPayment(new BigDecimal(1), "USD", "androidhub4you.com",
    PayPalPayment.PAYMENT_INTENT_SALE);
Intent intent = new Intent(MainActivity.this, PaymentActivity.class);
intent.putExtra(PaymentActivity.EXTRA_PAYMENT, thingToBuy);

startActivityForResult(intent, REQUEST_PAYPAL_PAYMENT);
```

8) 最后从 onActivityResult 获取支付响应

Chapter 159: Android PayPal Gateway Integration

Section 159.1: Setup PayPal in your android code

1) First go through Paypal Developer web site and create an application.

2) Now open your manifest file and give the below permissions

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3) And some required Activity and Services

```
<service
    android:name="compaypal.android.sdk.payments.PayPalService"
    android:exported="false" />
<activity android:name="compaypal.android.sdk.payments.PaymentActivity" />
<activity android:name="compaypal.android.sdk.payments.LoginActivity" />
<activity android:name="compaypal.android.sdk.payments.PaymentMethodActivity" />
<activity android:name="compaypal.android.sdk.payments.PaymentConfirmActivity" />
<activity android:name="compaypal.android.sdk.payments.PayPalFuturePaymentActivity" />
<activity android:name="compaypal.android.sdk.payments.FuturePaymentConsentActivity" />
<activity android:name="compaypal.android.sdk.payments.FuturePaymentInfoActivity" />
<activity
    android:name="io.card.payment.CardIOActivity"
    android:configChanges="keyboardHidden|orientation" />
<activity android:name="io.card.payment.DataEntryActivity" />
```

4) Open your Activity class and set Configuration for your app

```
//set the environment for production/sandbox/no netwrk
private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Now set client id from the Paypal developer account

```
private static final String CONFIG_CLIENT_ID = "PUT YOUR CLIENT ID";
```

6) Inside onCreate method call the Paypal service

```
Intent intent = new Intent(this, PayPalService.class);
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
startService(intent);
```

7) Now you are ready to make a payment just on button press call the Payment Activity

```
PayPalPayment thingToBuy = new PayPalPayment(new BigDecimal(1), "USD", "androidhub4you.com",
    PayPalPayment.PAYMENT_INTENT_SALE);
Intent intent = new Intent(MainActivity.this, PaymentActivity.class);
intent.putExtra(PaymentActivity.EXTRA_PAYMENT, thingToBuy);

startActivityForResult(intent, REQUEST_PAYPAL_PAYMENT);
```

8) And finally from the onActivityResult get the payment response

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_PAYPAL_PAYMENT) {
        if (resultCode == Activity.RESULT_OK) {
PaymentConfirmation confirm = data
                .getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
            if (confirm != null) {
                try {
                    System.out.println("Responseeee"+confirm);
Log.i("paymentExample", confirm.toJSONString().toString());
                }
                JSONObject jsonObj=new
JSONObject(confirm.toJSONString().toString());
                String paymentId=jsonObj.getJSONObject("response").getString("id");
                System.out.println("payment id:---"+paymentId);
Toast.makeText(getApplicationContext(), paymentId,
Toast.LENGTH_LONG).show();
            } catch (JSONException e) {
Log.e("paymentExample", "an extremely unlikely failure occurred: ", e);
        }
    }
} else if (resultCode == Activity.RESULT_CANCELED) {
    Log.i("paymentExample", "The user canceled.");
} else if (resultCode == PaymentActivity.RESULT_EXTRAS_INVALID) {
    Log.i("paymentExample", "An invalid Payment was submitted. Please see the
docs.");
}
}
}

```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_PAYPAL_PAYMENT) {
        if (resultCode == Activity.RESULT_OK) {
PaymentConfirmation confirm = data
                .getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
            if (confirm != null) {
                try {
                    System.out.println("Responseeee"+confirm);
Log.i("paymentExample", confirm.toJSONString().toString());
                }
                JSONObject jsonObj=new
JSONObject(confirm.toJSONString().toString());
                String paymentId=jsonObj.getJSONObject("response").getString("id");
                System.out.println("payment id:---"+paymentId);
Toast.makeText(getApplicationContext(), paymentId,
Toast.LENGTH_LONG).show();
            } catch (JSONException e) {
Log.e("paymentExample", "an extremely unlikely failure occurred: ", e);
        }
    }
} else if (resultCode == Activity.RESULT_CANCELED) {
    Log.i("paymentExample", "The user canceled.");
} else if (resultCode == PaymentActivity.RESULT_EXTRAS_INVALID) {
    Log.i("paymentExample", "An invalid Payment was submitted. Please see the
docs.");
}
}

```

第160章：可绘制对象

第160.1节：自定义可绘制对象

继承Drawable类并重写以下方法

```
public class IconDrawable extends Drawable {
    /**
     * 用于绘制形状的画笔
     */
    private Paint paint;
    /**
     * 要绘制在形状中心的图标可绘制对象
     */
    private Drawable icon;
    /**
     * 图标的期望宽度和高度
     */
    private int desiredIconHeight, desiredIconWidth;

    /**
     * Icon drawable 的公共构造函数
     *
     * @param icon      传入要绘制在中心的图标的可绘制对象
     * @param backgroundColor 形状的背景颜色
     */
    public IconDrawable(Drawable icon, int backgroundColor) {
        this.icon = icon;
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(backgroundColor);
        desiredIconWidth = 50;
        desiredIconHeight = 50;
    }

    @Override
    public void draw(Canvas canvas) {
        //如果我们将此可绘制对象设置为80dpX80dp的ImageView
        //getBounds将返回该尺寸，我们可以根据该宽度进行绘制。
        Rect bounds = getBounds();
        //以中心为原点，中心距离为半径绘制圆形
        canvas.drawCircle(bounds.centerX(), bounds.centerY(), bounds.centerX(), paint);
        //将图标可绘制对象的边界设置到形状的中心
        icon.setBounds(bounds.centerX() - (desiredIconWidth / 2), bounds.centerY() -
        (desiredIconHeight / 2), (bounds.centerX() - (desiredIconWidth / 2)) + desiredIconWidth,
        (bounds.centerY() - (desiredIconHeight / 2)) + desiredIconHeight);
        //在边界内绘制图标
        icon.draw(canvas);
    }

    @Override
    public void setAlpha(int alpha) {
        //为整个形状设置透明度
        paint.setAlpha(alpha);
    }

    @Override
    public void setColorFilter(ColorFilter colorFilter) {
        //为整个形状设置颜色滤镜
        paint.setColorFilter(colorFilter);
    }
}
```

Chapter 160: Drawables

Section 160.1: Custom Drawable

Extend your class with Drawable and override these methods

```
public class IconDrawable extends Drawable {
    /**
     * Paint for drawing the shape
     */
    private Paint paint;
    /**
     * Icon drawable to be drawn to the center of the shape
     */
    private Drawable icon;
    /**
     * Desired width and height of icon
     */
    private int desiredIconHeight, desiredIconWidth;

    /**
     * Public constructor for the Icon drawable
     *
     * @param icon      pass the drawable of the icon to be drawn at the center
     * @param backgroundColor background color of the shape
     */
    public IconDrawable(Drawable icon, int backgroundColor) {
        this.icon = icon;
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(backgroundColor);
        desiredIconWidth = 50;
        desiredIconHeight = 50;
    }

    @Override
    public void draw(Canvas canvas) {
        //if we are setting this drawable to a 80dpX80dp imageview
        //getBounds will return that measurements,we can draw according to that width.
        Rect bounds = getBounds();
        //drawing the circle with center as origin and center distance as radius
        canvas.drawCircle(bounds.centerX(), bounds.centerY(), bounds.centerX(), paint);
        //set the icon drawable's bounds to the center of the shape
        icon.setBounds(bounds.centerX() - (desiredIconWidth / 2), bounds.centerY() -
        (desiredIconHeight / 2), (bounds.centerX() - (desiredIconWidth / 2)) + desiredIconWidth,
        (bounds.centerY() - (desiredIconHeight / 2)) + desiredIconHeight);
        //draw the icon to the bounds
        icon.draw(canvas);
    }

    @Override
    public void setAlpha(int alpha) {
        //sets alpha to your whole shape
        paint.setAlpha(alpha);
    }

    @Override
    public void setColorFilter(ColorFilter colorFilter) {
        //sets color filter to your whole shape
        paint.setColorFilter(colorFilter);
    }
}
```

```

}
@Override
public int getOpacity() {
    //返回形状所需的不透明度
    return PixelFormat.TRANSLUCENT;
}

```

在布局中声明一个 ImageView

```
<ImageView
    android:layout_width="80dp"
    android:id="@+id/imageView"
    android:layout_height="80dp" />
```

将自定义的 drawable 设置到 ImageView

```
IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(this,R.color.pink_300));
imageView.setImageDrawable(iconDrawable);
```

截图



第160.2节：为可绘制对象着色

可绘制对象可以被着色为某种颜色。这对于支持应用程序中的不同主题以及减少可绘制资源文件的数量非常有用。

在SDK 21及以上版本使用框架API：

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

在SDK 4及以上版本使用android.support.v4库：

```
//加载未着色的资源
final Drawable drawableRes = ContextCompat.getDrawable(context, R.drawable.ic_launcher);
//使用兼容库包装它，以便可以修改
Drawable tintedDrawable = DrawableCompat.wrap(drawableRes);
//应用彩色滤镜
DrawableCompat.setTint(tintedDrawable, Color.WHITE);
//此时你可以像平常一样使用tintedDrawable
//(drawableRes可以丢弃)

//注意：如果你的原始drawableRes正在被使用（例如它是调用'getBackground()'方法的结果），那么此时你
//仍然需要替换背景。setTint不会修改drawableRes指向的实例，而是创建了一个新的drawable实例
```

请注意，int颜色并不是指颜色资源，但你不局限于使用'Color'类中定义的颜色

```

}
@Override
public int getOpacity() {
    //give the desired opacity of the shape
    return PixelFormat.TRANSLUCENT;
}

```

Declare a ImageView in your layout

```
<ImageView
    android:layout_width="80dp"
    android:id="@+id/imageView"
    android:layout_height="80dp" />
```

Set your custom drawable to the ImageView

```
IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(this,R.color.pink_300));
imageView.setImageDrawable(iconDrawable);
```

Screenshot



Section 160.2: Tint a drawable

A drawable can be tinted a certain color. This is useful for supporting different themes within your application, and reducing the number of drawable resource files.

Using framework APIs on SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Using android.support.v4 library on SDK 4+:

```
//Load the untinted resource
final Drawable drawableRes = ContextCompat.getDrawable(context, R.drawable.ic_launcher);
//Wrap it with the compatibility library so it can be altered
Drawable tintedDrawable = DrawableCompat.wrap(drawableRes);
//Apply a coloured tint
DrawableCompat.setTint(tintedDrawable, Color.WHITE);
//At this point you may use the tintedDrawable just as you usually would
//(and drawableRes can be discarded)
```

```
//NOTE: If your original drawableRes was in use somewhere (i.e. it was the result of
//a call to a `getBackground()` method then at this point you still need to replace
//the background. setTint does *not* alter the instance that drawableRes points to,
//but instead creates a new drawable instance
```

Please note that **int** color is **not** referring to a color Resource, however you are not limited to those colours defined

当你在XML中定义了一个颜色并想使用时，你必须先获取它的值。

你可以使用以下方法替换Color.WHITE的用法

针对较旧的API：

```
getResources().getColor(R.color.your_color);
```

或者针对较新的目标：

```
ContextCompat.getColor(context, R.color.your_color);
```

第160.3节：圆形视图

对于圆形视图（此处为TextView），在drawable文件夹中创建一个名为round_view.xml的drawable：

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#FAA23C" />
    <stroke android:color="#FFF" android:width="2dp" />
</shape>
```

将该drawable分配给视图：

```
<TextView
    android:id="@+id/game_score"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:background="@drawable/round_score"
    android:padding="6dp"
    android:text="100"
    android:textColor="#fff"
    android:textSize="20sp"
    android:textStyle="bold"
    android:gravity="center" />
```

现在它应该看起来像橙色圆圈：



第160.4节：制作带圆角的视图

在 drawable 文件夹中创建名为 custom_rectangle.xml 的 drawable 文件：

in the 'Color' class. When you have a colour defined in your XML which you want to use you must just first get it's value.

You can replace usages of Color.WHITE using the methods below

When targeting older API's:

```
getResources().getColor(R.color.your_color);
```

Or on newer targets:

```
ContextCompat.getColor(context, R.color.your_color);
```

Section 160.3: Circular View

For a circular View (in this case TextView) create a drawable **round_view.xml** in **drawable** folder:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#FAA23C" />
    <stroke android:color="#FFF" android:width="2dp" />
</shape>
```

Assign the drawable to the View:

```
<TextView
    android:id="@+id/game_score"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:background="@drawable/round_score"
    android:padding="6dp"
    android:text="100"
    android:textColor="#fff"
    android:textSize="20sp"
    android:textStyle="bold"
    android:gravity="center" />
```

Now it should look like the orange circle:



Section 160.4: Make View with rounded corners

Create **drawable** file named with **custom_rectangle.xml** in **drawable** folder:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <solid android:color="@android:color/white" />

    <corners android:radius="10dip" />

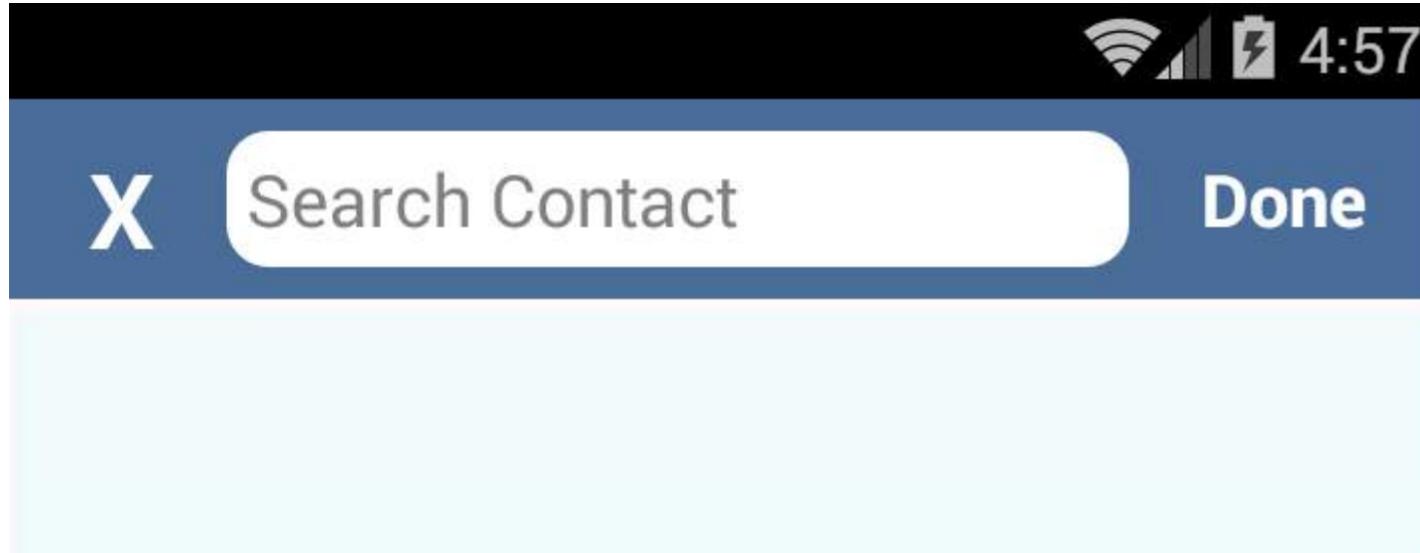
    <stroke
        android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

现在将矩形背景应用到视图（View）上：

```
mView.setBackground(R.drawable.custom_rectangle);
```

参考截图：



```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <solid android:color="@android:color/white" />

    <corners android:radius="10dip" />

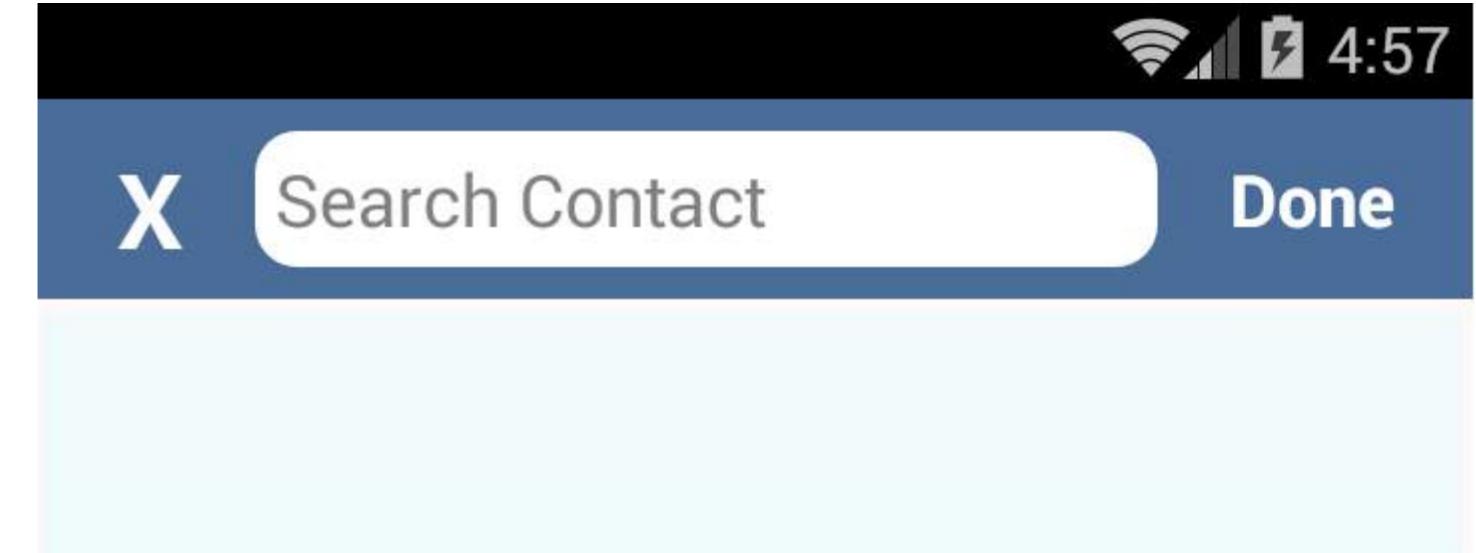
    <stroke
        android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

Now apply **rectangle background** on View:

```
mView.setBackground(R.drawable.custom_rectangle);
```

Reference screenshot:



第161章：TransitionDrawable

第161.1节：使用TransitionDrawable动画视图背景颜色（切换颜色）

```
public void setCardColorTran(View view) {  
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};  
    TransitionDrawable trans = new TransitionDrawable(color);  
    if(Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.JELLY_BEAN) {  
        view.setBackgroundDrawable(trans);  
    } else {  
        view.setBackground(trans);  
    }  
    trans.startTransition(5000);  
}
```

第161.2节：在两张图像之间添加过渡或交叉淡入淡出

步骤1：在XML中创建一个过渡drawable

将此文件transition.xml保存到项目的res/drawable文件夹中。

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:drawable="@drawable/image1"/>  
    <item android:drawable="@drawable/image2"/>  
</transition>
```

image1和image2是我们想要过渡的两张图片，它们也应该放在你的res/drawable文件夹中。

步骤2：在你的XML布局中为ImageView添加代码以显示上述drawable。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    tools:context=".MainActivity" >  
  
    <ImageView  
        android:id="@+id/image_view"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:src="@drawable/image1" />  
  
</LinearLayout>
```

步骤3：在Activity的onCreate()方法中访问XML过渡drawable，并在onClick()事件中启动过渡。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    imageView = (ImageView) findViewById(R.id.image_view);  
    transitionDrawable = (TransitionDrawable)
```

Chapter 161: TransitionDrawable

Section 161.1: Animate views background color (switch-color) with TransitionDrawable

```
public void setCardColorTran(View view) {  
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};  
    TransitionDrawable trans = new TransitionDrawable(color);  
    if(Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.JELLY_BEAN) {  
        view.setBackgroundDrawable(trans);  
    } else {  
        view.setBackground(trans);  
    }  
    trans.startTransition(5000);  
}
```

Section 161.2: Add transition or Cross-fade between two images

Step 1: Create a transition drawable in XML

Save this file transition.xml in res/drawable folder of your project.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:drawable="@drawable/image1"/>  
    <item android:drawable="@drawable/image2"/>  
</transition>
```

The image1 and image2 are the two images that we want to transition and they should be put in your res/drawable folder too.

Step 2: Add code for ImageView in your XML layout to display the above drawable.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    tools:context=".MainActivity" >  
  
    <ImageView  
        android:id="@+id/image_view"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:src="@drawable/image1" />  
  
</LinearLayout>
```

Step 3: Access the XML transition drawable in onCreate() method of your Activity and start transition in onClick() event.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    imageView = (ImageView) findViewById(R.id.image_view);  
    transitionDrawable = (TransitionDrawable)
```

```
ContextCompat.getDrawable(this, R.drawable.transition);

birdImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View view) {
        birdImageView.setImageDrawable(transitionDrawable);
        transitionDrawable.startTransition(1000);
    }
});
```

```
ContextCompat.getDrawable(this, R.drawable.transition);

birdImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View view) {
        birdImageView.setImageDrawable(transitionDrawable);
        transitionDrawable.startTransition(1000);
    }
});
```

第162章：矢量Drawable

参数	详细信息
<vector>	用于定义矢量图形
<group>	定义一组路径或子组，以及变换信息。变换是在与视口相同的坐标系中定义的。变换的应用顺序为缩放、旋转，然后平移。
<path>	定义要绘制的路径。
<clip-path>	定义路径作为当前裁剪路径。注意，裁剪路径仅应用于当前组及其子元素。

顾名思义，矢量可绘制对象基于矢量图形。矢量图形是一种使用几何形状描述图形元素的方式。这使您可以基于 XML 矢量图形创建可绘制对象。现在无需为 mdpi、hdpi、xhdpi 等设计不同尺寸的图像。使用矢量可绘制对象，您只需创建一次 XML 文件格式的图像，即可针对所有 dpi 和不同设备进行缩放。这不仅节省了空间，还简化了维护。

第 162.1 节：将 SVG 文件导入为 VectorDrawable

您可以在 Android Studio 中将SVG文件导入为VectorDrawable，操作步骤如下：

在res文件夹上“右键点击”，选择new > Vector Asset。

Chapter 162: Vector Drawables

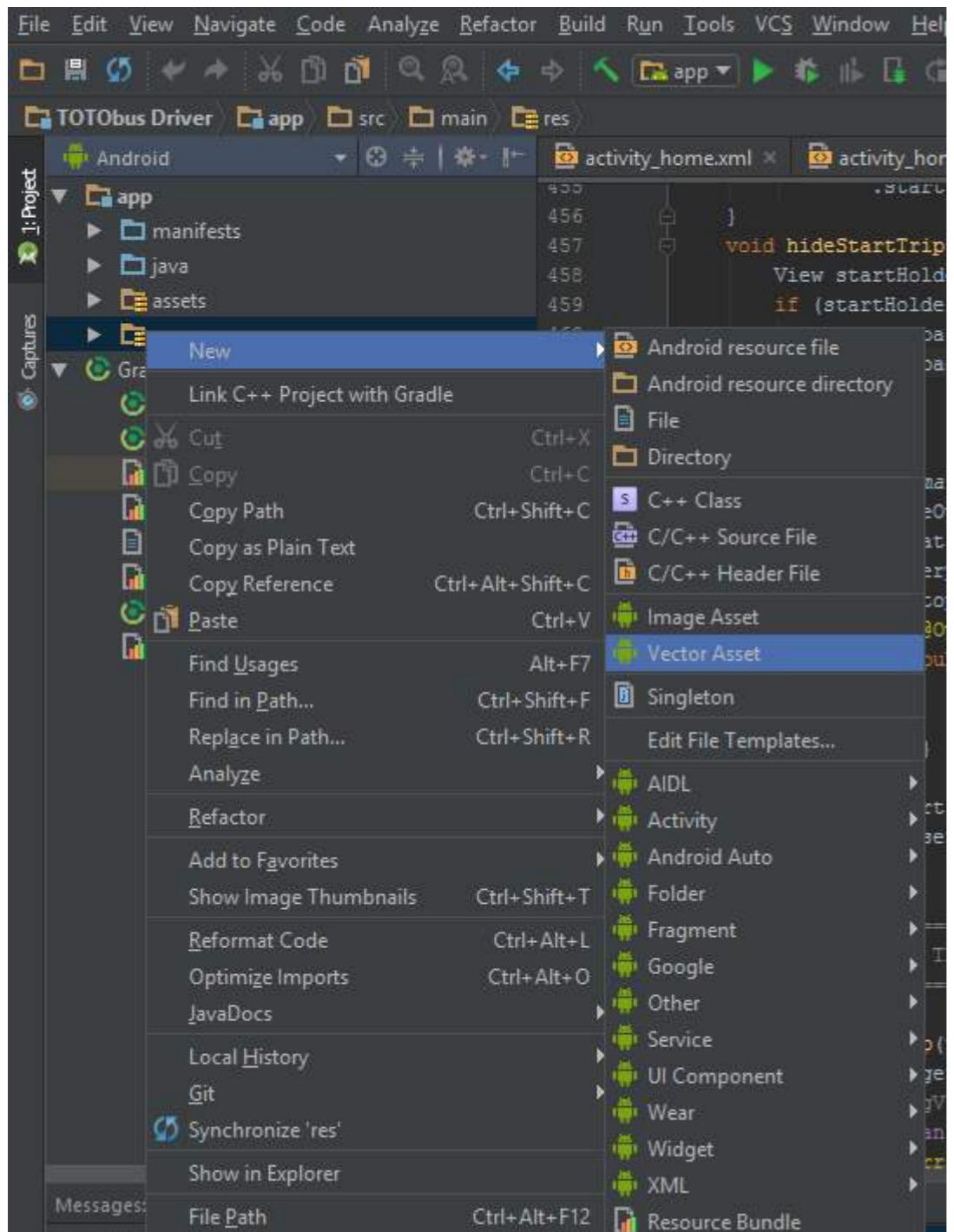
Parameter	Details
<vector>	Used to define a vector drawable
<group>	Defines a group of paths or subgroups, plus transformation information. The transformations are defined in the same coordinates as the viewport. And the transformations are applied in the order of scale, rotate then translate.
<path>	Defines paths to be drawn.
<clip-path>	Defines path to be the current clip. Note that the clip path only apply to the current group and its children.

As the name implies, vector drawables are based on vector graphics. Vector graphics are a way of describing graphical elements using geometric shapes. This lets you create a drawable based on an XML vector graphic. Now there is no need to design different size image for mdpi, hdpi, xhdpi and etc. With Vector Drawable you need to create image only once as an xml file and you can scale it for all dpi and for different devices. This also not save space but also simplifies maintenance.

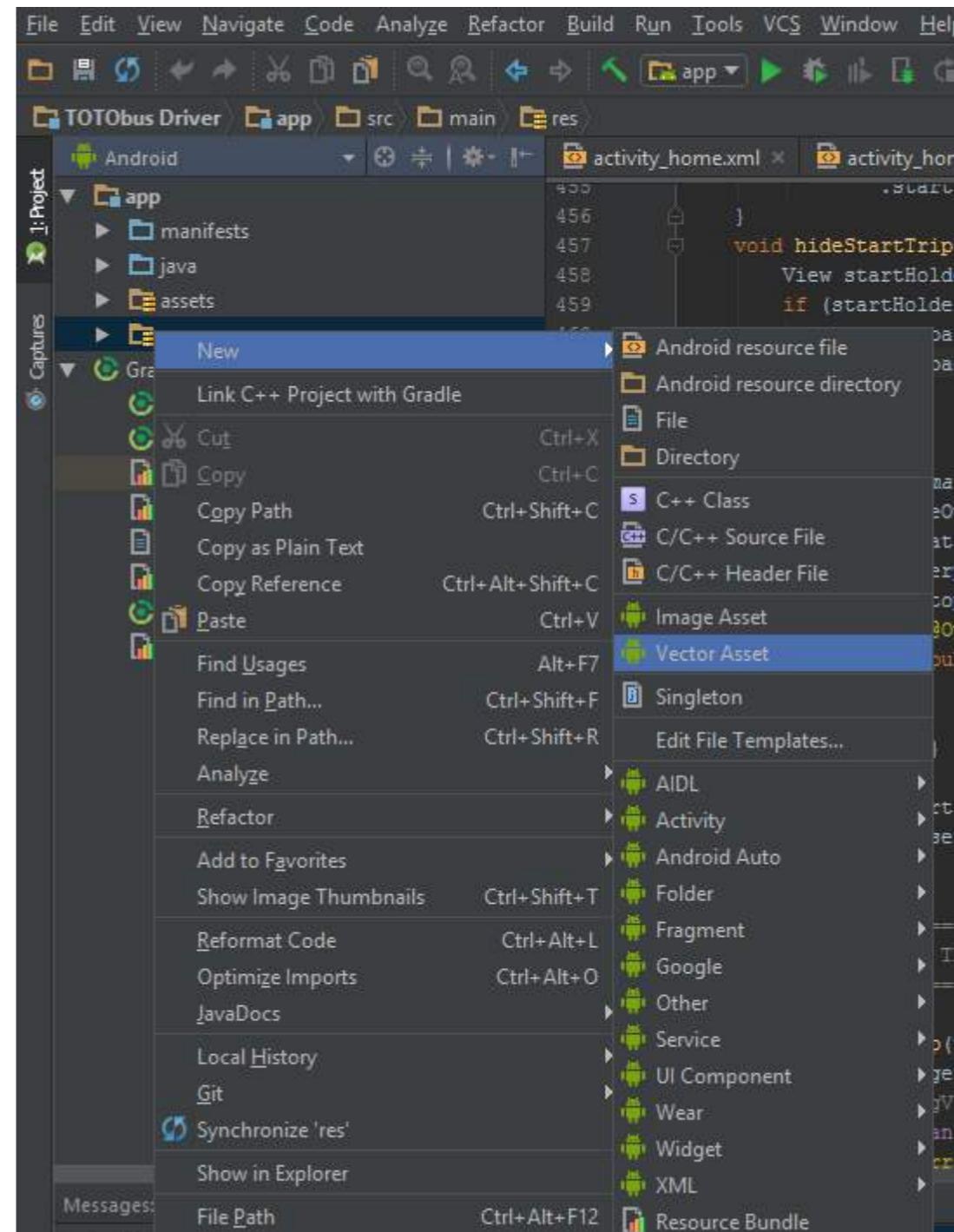
Section 162.1: Importing SVG file as VectorDrawable

You can import an SVG file as a VectorDrawable in Android Studio, follow these steps :

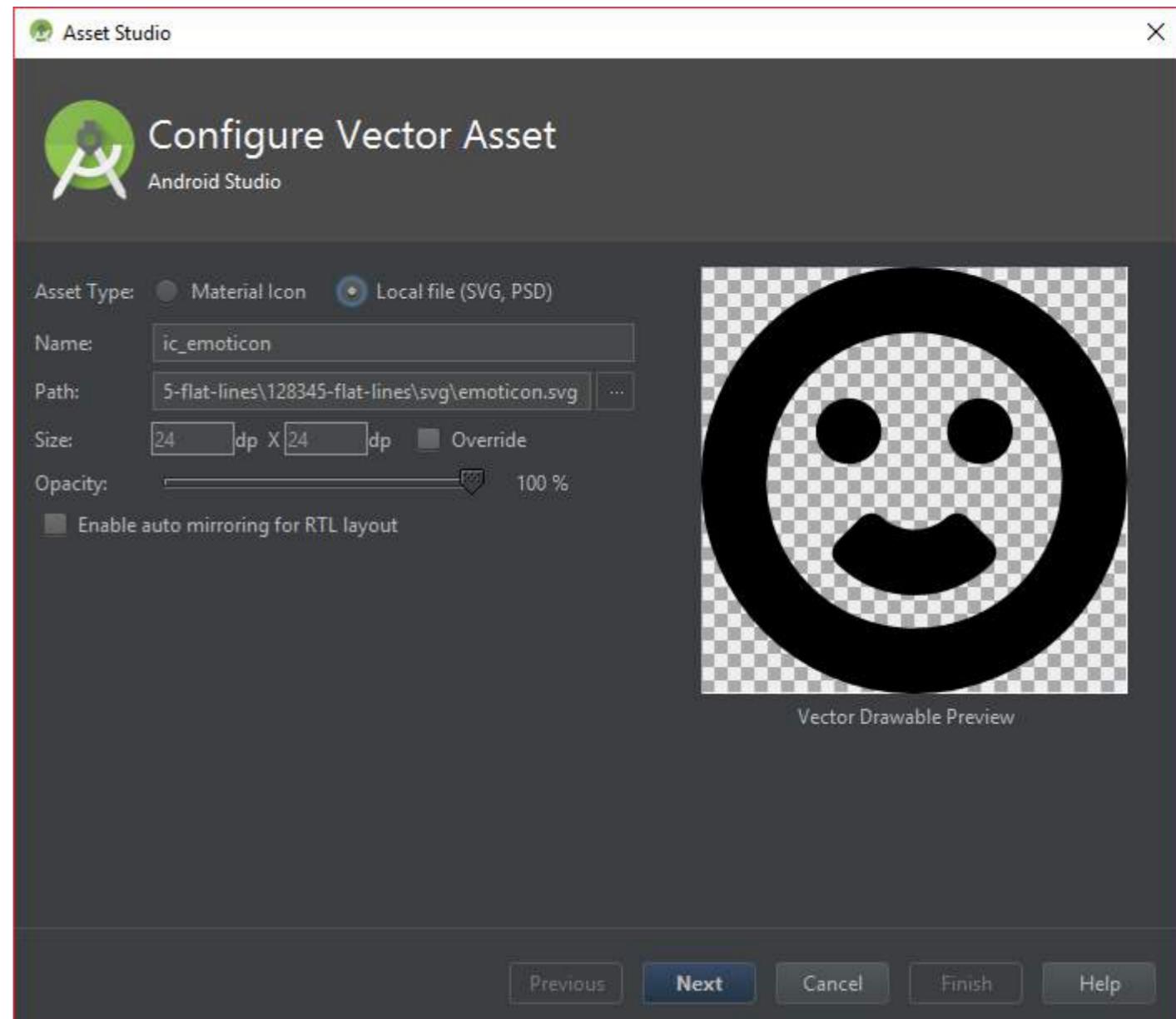
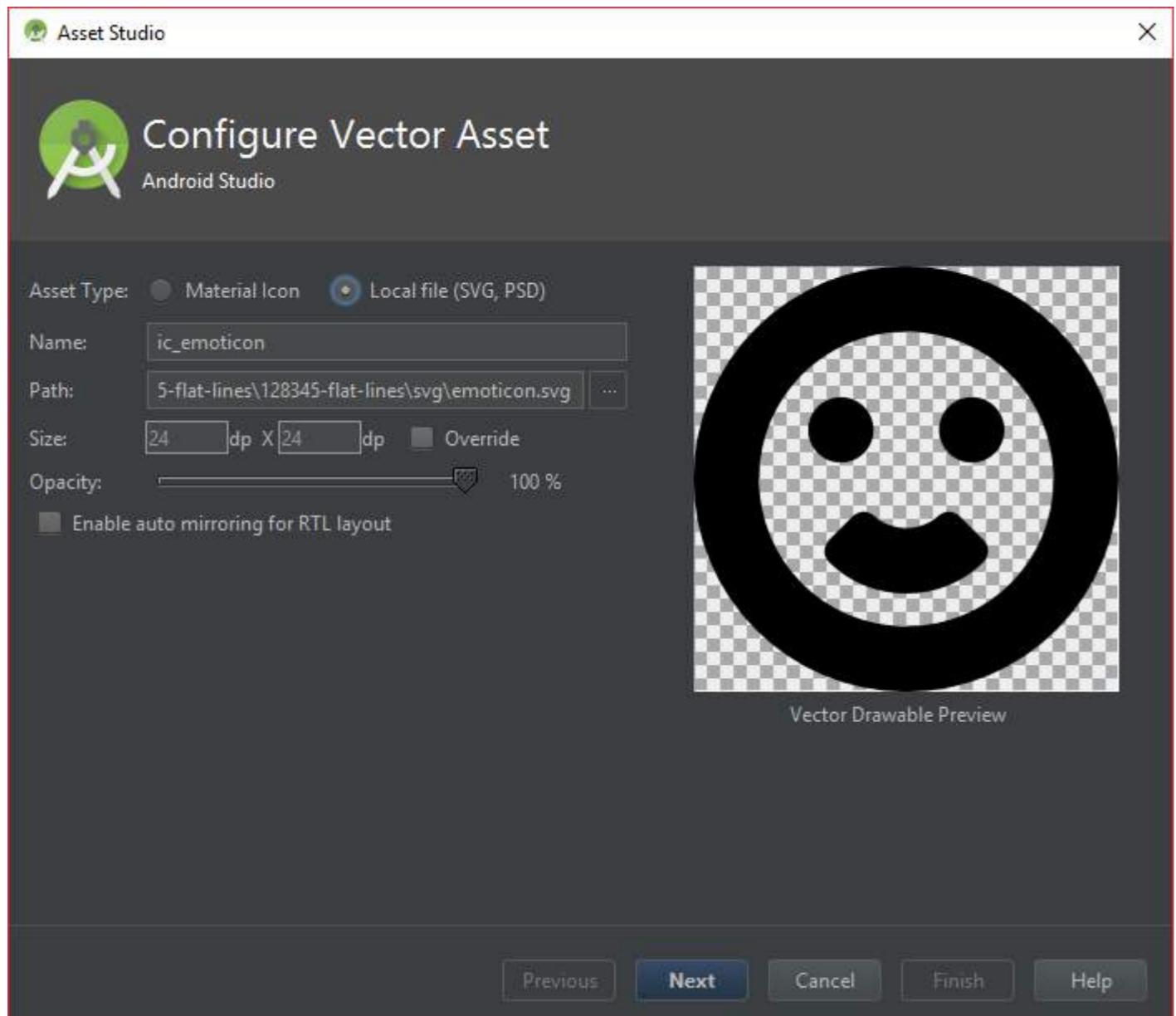
"Right-click" on the res folder and select **new > Vector Asset**.



选择本地文件选项并浏览到您的.svg文件。根据您的喜好更改选项，然后点击下一步。完成。



Select the **Local File** option and browse to your .svg file. Change the options to your liking and hit next. Done.



第162.2节：VectorDrawable使用示例

这里是一个我们实际在AppCompat中使用的矢量资源示例：

res/drawable/ic_search.xml

```
<vector xmlns:android="..." android:width="24dp" android:height="24dp" android:viewportWidth="24.0" android:viewportHeight="24.0" android:tint="?attr/colorControlNormal">

<path android:pathData="..." android:fillColor="@android:color/white"/>
```

使用此可绘制对象，示例ImageView声明如下：

<ImageView

Section 162.2: VectorDrawable Usage Example

Here's an example vector asset which we're actually using in AppCompat:

res/drawable/ic_search.xml

```
<vector xmlns:android="..." android:width="24dp" android:height="24dp" android:viewportWidth="24.0" android:viewportHeight="24.0" android:tint="?attr/colorControlNormal">

<path android:pathData="..." android:fillColor="@android:color/white"/>
```

Using this drawable, an example ImageView declaration would be:

<ImageView

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/ic_search"/>
```

您也可以在运行时设置它：

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

相同的属性和调用也适用于ImageButton。

第162.3节：VectorDrawable XML示例

这是一个简单的VectorDrawable，位于这个vectordrawable.xml文件中。

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/ic_search"/>
```

You can also set it at run-time:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

The same attribute and calls work for ImageButton too.

Section 162.3: VectorDrawable xml example

Here is a simple VectorDrawable in this **vectordrawable.xml** file.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

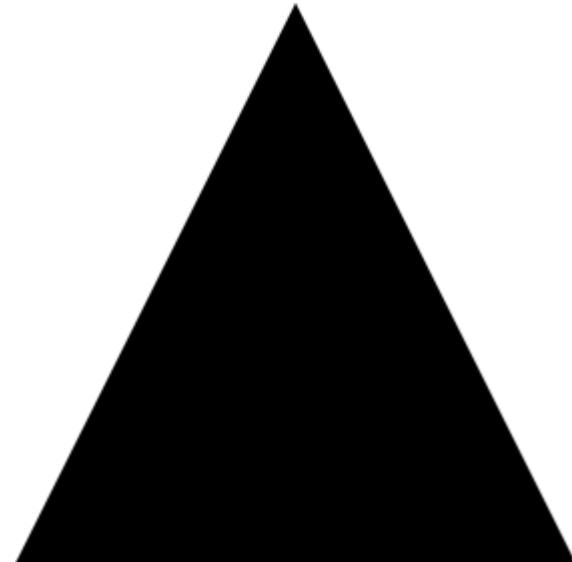
第163章 : VectorDrawable和AnimatedVectorDrawable

第163.1节 : 基础矢量图形 (VectorDrawable)

A VectorDrawable 应至少包含一个定义形状的<path>标签

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportWidth="24.0"  
    android:viewportHeight="24.0">  
    <path  
        android:fillColor="#FF000000"  
        android:pathData="M0,24 112,-24 112,24 z"/>
```

这将生成一个黑色三角形：



第163.2节 : <group>标签

A <group> 标签允许调整一个或多个VectorDrawable元素的缩放、旋转和位置：

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportWidth="24.0"  
    android:viewportHeight="24.0">  
    <path  
        android:pathData="M0,0 h4 v4 h-4 z"  
        android:fillColor="#FF000000"/>  
  
    <group  
        android:name="middle square group"  
        android:translateX="10"  
        android:translateY="10"  
        android:rotation="45">  
        <path
```

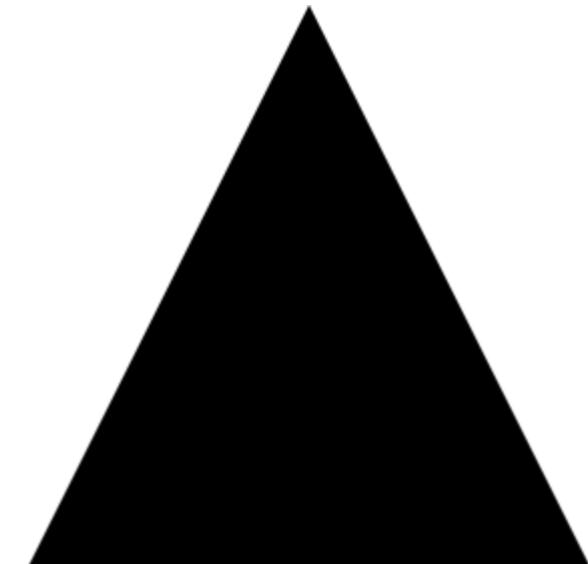
Chapter 163: VectorDrawable and AnimatedVectorDrawable

Section 163.1: Basic VectorDrawable

A VectorDrawable should consist of at least one <path> tag defining a shape

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportWidth="24.0"  
    android:viewportHeight="24.0">  
    <path  
        android:fillColor="#FF000000"  
        android:pathData="M0,24 112,-24 112,24 z"/>  
/>
```

This would produce a black triangle:



Section 163.2: <group> tags

A <group> tag allows the scaling, rotation, and position of one or more elements of a VectorDrawable to be adjusted:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportWidth="24.0"  
    android:viewportHeight="24.0">  
    <path  
        android:pathData="M0,0 h4 v4 h-4 z"  
        android:fillColor="#FF000000"/>  
  
    <group  
        android:name="middle square group"  
        android:translateX="10"  
        android:translateY="10"  
        android:rotation="45">  
        <path
```

```

        android:pathData="M0,0 h4 v4 h-4 z"
        android:fillColor="#FF000000" />
    </group>

    <group
        android:name="last square group"
        android:translateX="18"
        android:translateY="18"
        android:scaleX="1.5">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000" />
    </group>
</vector>

```

上面的示例代码包含三个相同的<path>标签，均描述黑色方块。第一个方块未做调整。第二个方块被包裹在一个<group>标签中，该标签将其移动并旋转45°。第三个方块被包裹在一个<group>标签中，该标签将其移动并水平拉伸50%。结果如下：



一个<group>标签可以包含多个<path>和<clip-path>标签。它甚至可以包含另一个<group>标签。

第163.3节：基本的AnimatedVectorDrawable

一个AnimatedVectorDrawable至少需要3个组件：

- 一个将被操作的VectorDrawable
- 一个定义要更改的属性及其方式的objectAnimatorAnimatedVectorDra
- wable本身，将objectAnimator与VectorDrawable连接以创建动画

以下创建了一个颜色从黑色过渡到红色的三角形。

矢量绘图 (VectorDrawable) ，文件名：triangle_vector_drawable.xml

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">

    <path

```

```

        android:pathData="M0,0 h4 v4 h-4 z"
        android:fillColor="#FF000000" />
    </group>

    <group
        android:name="last square group"
        android:translateX="18"
        android:translateY="18"
        android:scaleX="1.5">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000" />
    </group>
</vector>

```

The example code above contains three identical <path> tags, all describing black squares. The first square is unadjusted. The second square is wrapped in a <group> tag which moves it and rotates it by 45°. The third square is wrapped in a <group> tag which moves it and stretches it horizontally by 50%. The result is as follows:



A <group> tag can contain multiple <path> and <clip-path> tags. It can even contain another <group>.

Section 163.3: Basic AnimatedVectorDrawable

An AnimatedVectorDrawable requires at least 3 components:

- A VectorDrawable which will be manipulated
- An objectAnimator which defines what property to change and how
- The AnimatedVectorDrawable itself which connects the objectAnimator to the VectorDrawable to create the animation

The following creates a triangle that transitions its color from black to red.

The VectorDrawable, filename: triangle_vector_drawable.xml

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">

    <path

```

```
    android:name="triangle"
    android:fillColor="@android:color/black"
    android:pathData="M0,24 l12,-24 l12,24 z" />
```

对象动画 (objectAnimator) , 文件名 : color_change_animator.xml

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light" />
```

动画矢量绘图 (AnimatedVectorDrawable) , 文件名 : triangle_animated_vector.xml

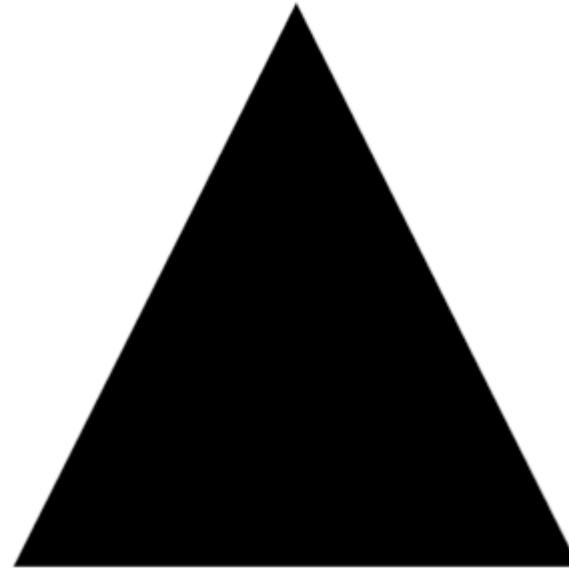
```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/triangle_vector_drawable" >

    <target
        android:animation="@animator/color_change_animator"
        android:name="triangle"/>

</animated-vector>
```

请注意, **<target>** 指定了 `android:name="triangle"`, 这与 **VectorDrawable** 中的 `<path>` 相匹配。一个 VectorDrawable 可能包含多个元素, `android:name` 属性用于定义被定位的元素。

结果 :



第163.4节：使用描边

使用SVG描边可以更容易地创建具有统一描边长度的矢量图形, 符合Material Design设计指南：

一致的描边粗细是统一整体系统图标系列的关键。所有

```
    android:name="triangle"
    android:fillColor="@android:color/black"
    android:pathData="M0,24 l12,-24 l12,24 z" />
```

```
</vector>
```

The objectAnimator, filename: color_change_animator.xml

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light" />
```

The AnimatedVectorDrawable, filename: triangle_animated_vector.xml

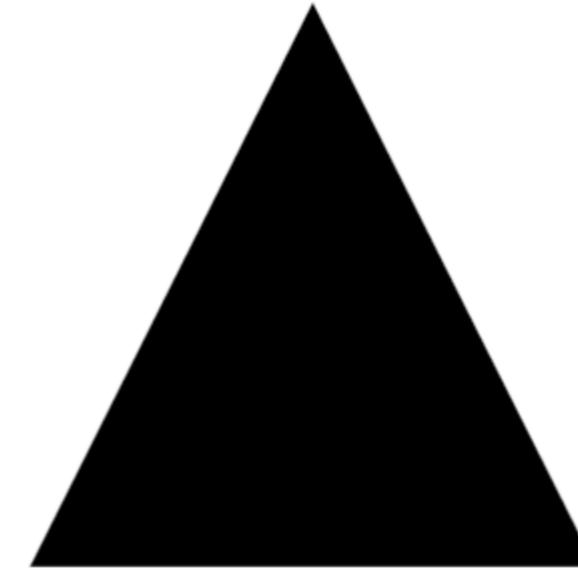
```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/triangle_vector_drawable" >

    <target
        android:animation="@animator/color_change_animator"
        android:name="triangle"/>

</animated-vector>
```

Note that the **<target>** specifies `android:name="triangle"` which matches the `<path>` in the VectorDrawable. A VectorDrawable may contain multiple elements and the `android:name` property is used to define which element is being targeted.

Result:



Section 163.4: Using Strokes

Using SVG stroke makes it easier to create a Vector drawable with unified stroke length, as per [Material Design guidelines](#):

Consistent stroke weights are key to unifying the overall system icon family. Maintain a 2dp width for all

描边实例，包括曲线、角度以及内外描边，都保持2dp宽度。

例如，以下是使用描边创建“加号”符号的方法：

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportHeight="24.0"  
    android:viewportWidth="24.0">  
    <path  
        android:fillColor="#FF000000"  
        android:strokeColor="#F000"  
        android:strokeWidth="2"  
        android:pathData="M12,0 V24 M0,12 H24" />
```

- strokeColor 定义描边的颜色。
- strokeWidth 定义描边的宽度（单位为dp，本例中为2dp，符合指南建议）。
- pathData 是我们描述SVG图像的地方：
 - M12, 0 将“光标”移动到位置12,0
 - V24 创建一条垂直线到位置12,24

等等，参见SVG文档以及w3schools的这个有用的“SVG路径”教程，了解更多关于具体路径命令的信息。

结果，我们得到了这个简洁的加号：



这对于创建AnimatedVectorDrawable尤其有用，因为你现在操作的是单一描边且长度统一，而不是复杂的路径。

stroke instances, including curves, angles, and both interior and exterior strokes.

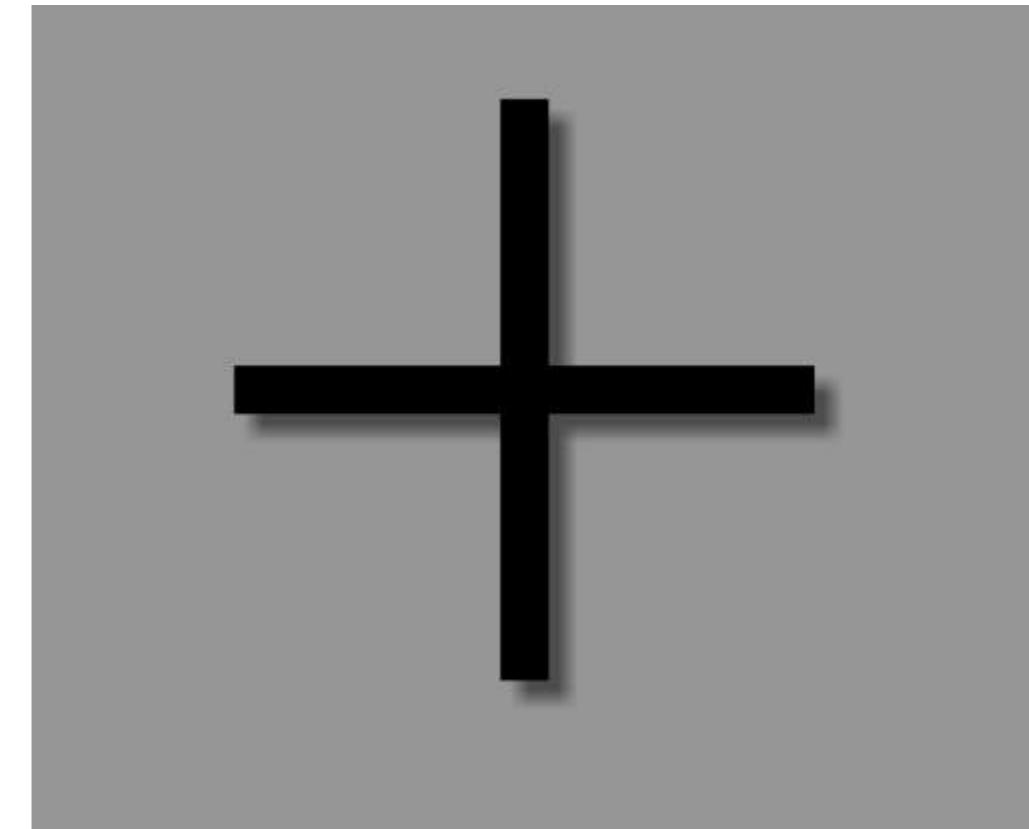
So, for example, this is how you would create a "plus" sign using strokes:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="24dp"  
    android:height="24dp"  
    android:viewportHeight="24.0"  
    android:viewportWidth="24.0">  
    <path  
        android:fillColor="#FF000000"  
        android:strokeColor="#F000"  
        android:strokeWidth="2"  
        android:pathData="M12,0 V24 M0,12 H24" />  
    </vector>
```

- strokeColor defines the color of the stroke.
- strokeWidth defines the width (in dp) of the stroke (2dp in this case, as suggested by the guidelines).
- pathData is where we describe our SVG image:
 - M12, 0 moves the "cursor" to the position 12,0
 - V24 creates a vertical line to the position 12, 24

etc., see SVG documentation and this useful ["SVG Path" tutorial from w3schools](#) to learn more about the specific path commands.

As a result, we got this no-frills plus sign:



This is **especially useful** for creating an AnimatedVectorDrawable, since you are now operating with a single stroke with an unified length, instead of an otherwise complicated path.

第163.5节：使用 <clip-path>

A <clip-path> 定义了一个形状，作为一个窗口，只允许 <path> 的部分内容在 <clip-path> 形状内显示，其他部分被裁剪掉。

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <clip-path
        android:name="方形裁剪路径"
        android:pathData="M6,6 h12 v12 h-12 z"/>
    <path
        android:name="三角形"
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

```

在这种情况下，<path> 生成了一个黑色三角形，但 <clip-path> 定义了一个较小的方形，只允许三角形的一部分显示出 来：



Section 163.5: Using <clip-path>

A <clip-path> defines a shape which acts as a window, only allowing parts of a <path> to show if they are within the <clip-path> shape and cutting off the rest.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <clip-path
        android:name="square clip path"
        android:pathData="M6,6 h12 v12 h-12 z"/>
    <path
        android:name="triangle"
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

```

</vector>

In this case the <path> produces a black triangle, but the <clip-path> defines a smaller square shape, only allowing part of the triangle to show through:



第163.6节：通过AppCompat实现矢量兼容性

在 build.gradle 中有一些前提条件，使矢量图能够一直支持到API 7（针对VectorDrawables）和API 13（针对AnimatedVectorDrawables，当前有一些限制）：

```
//Build Tools 必须是24及以上版本
buildToolsVersion '24.0.0'

defaultConfig {
    vectorDrawables.useSupportLibrary = true
    generatedDensities = []
    aaptOptions {
        additionalParameters "--no-version-vectors"
    }
}

dependencies {
```

Section 163.6: Vector compatibility through AppCompat

A few pre-requisites in the build.gradle for vectors to work all the way down to API 7 for VectorDrawables and API 13 for AnimatedVectorDrawables (with some caveats currently):

```
//Build Tools has to be 24+
buildToolsVersion '24.0.0'

defaultConfig {
    vectorDrawables.useSupportLibrary = true
    generatedDensities = []
    aaptOptions {
        additionalParameters "--no-version-vectors"
    }
}

dependencies {
```

```
compile 'com.android.support:appcompat-v7:24.1.1'  
}
```

在你的layout.xml中：

```
<ImageView  
    android:id="@+id/android"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    appCompat:src="@drawable/vector_drawable"  
    android:contentDescription="@null" />
```

```
compile 'com.android.support:appcompat-v7:24.1.1'  
}
```

In your layout.xml:

```
<ImageView  
    android:id="@+id/android"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    appCompat:src="@drawable/vector_drawable"  
    android:contentDescription="@null" />
```

第164章：使用Cling库在Android中进行端口映射

第164.1节：映射NAT端口

```
String myIp = getIpAddress();
int port = 55555;

// 创建一个端口映射配置，包含外部/内部端口、内部主机IP、协议和可选描述
PortMapping[] desiredMapping = new PortMapping[2];
desiredMapping[0] = new PortMapping(port, myIp, PortMapping.Protocol.TCP);
desiredMapping[1] = new PortMapping(port, myIp, PortMapping.Protocol.UDP);

// 启动UPnP服务
UpnpService upnpService = new UpnpServiceImpl(new AndroidUpnpServiceConfiguration());
RegistryListener registryListener = new PortMappingListener(desiredMapping);
upnpService.getRegistry().addListener(registryListener);
upnpService.getControlPoint().search();

// 获取本地IP的方法
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
            .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
                .nextElement();
            枚举<InetAddress> enumInetAddress = networkInterface
                .getInetAddresses();
            while (enumInetAddress.hasMoreElements()) {
                InetAddress inetAddress = enumInetAddress.nextElement();

                if (inetAddress.isSiteLocalAddress()) {
                    ip += inetAddress.getHostAddress();
                }
            }
        }
    } catch (SocketException e) {
        // TODO 自动生成的 catch 块
        e.printStackTrace();
        ip += "Something Wrong! " + e.toString() + "\n";
    }
    return ip;
}
```

第164.2节：为您的Android项目添加Cling支持

build.gradle

```
repositories {
    maven { url 'http://4thline.org/m2' }
}

dependencies {
```

Chapter 164: Port Mapping using Cling library in Android

Section 164.1: Mapping a NAT port

```
String myIp = getIpAddress();
int port = 55555;

//creates a port mapping configuration with the external/internal port, an internal host IP, the
//protocol and an optional description
PortMapping[] desiredMapping = new PortMapping[2];
desiredMapping[0] = new PortMapping(port, myIp, PortMapping.Protocol.TCP);
desiredMapping[1] = new PortMapping(port, myIp, PortMapping.Protocol.UDP);

//starting the UPnP service
UpnpService upnpService = new UpnpServiceImpl(new AndroidUpnpServiceConfiguration());
RegistryListener registryListener = new PortMappingListener(desiredMapping);
upnpService.getRegistry().addListener(registryListener);
upnpService.getControlPoint().search();

//method for getting local ip
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
            .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
                .nextElement();
            Enumeration<InetAddress> enumInetAddress = networkInterface
                .getInetAddresses();
            while (enumInetAddress.hasMoreElements()) {
                InetAddress inetAddress = enumInetAddress.nextElement();

                if (inetAddress.isSiteLocalAddress()) {
                    ip += inetAddress.getHostAddress();
                }
            }
        }
    } catch (SocketException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        ip += "Something Wrong! " + e.toString() + "\n";
    }
    return ip;
}
```

Section 164.2: Adding Cling Support to your Android Project

build.gradle

```
repositories {
    maven { url 'http://4thline.org/m2' }
}

dependencies {
```

```
// Cling
compile 'org.fourthline.cling:cling-support:2.1.0'

// Cling 所需的其他依赖
compile 'org.eclipse.jetty:jetty-server:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-servlet:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-client:8.1.18.v20150929'
compile 'org.slf4j:slf4j-jdk14:1.7.14'

}'
```

```
// Cling
compile 'org.fourthline.cling:cling-support:2.1.0'

// Other dependencies required by Cling
compile 'org.eclipse.jetty:jetty-server:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-servlet:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-client:8.1.18.v20150929'
compile 'org.slf4j:slf4j-jdk14:1.7.14'

}'
```

第165章：创建覆盖（始终置顶）窗口

第165.1节：弹出覆盖层

为了将您的视图置于所有应用程序之上，您必须将视图分配给相应的窗口管理器。为此，您需要系统警报权限，可以通过在清单文件中添加以下行来请求该权限：

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

注意：如果您的应用被销毁，您的视图将从窗口管理器中移除。因此，最好通过前台服务创建视图并将其分配给窗口管理器。

将视图分配给窗口管理器

您可以按如下方式获取窗口管理器实例：

```
WindowManager mWindowManager = (WindowManager) mContext.getSystemService(Context.WINDOW_SERVICE);
```

为了定义视图的位置，您需要创建一些布局参数，如下所示：

```
WindowManager.LayoutParams mLayoutParams = new WindowManager.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_PHONE,
    WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON,
    PixelFormat.TRANSLUCENT);
mLayoutParams.gravity = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;
```

现在，您可以将视图和创建的布局参数一起分配给窗口管理器实例，如下所示：

```
mWindowManager.addView(yourView, mLayoutParams);
```

完成！您的视图已成功置于所有其他应用程序之上。

注意：您的视图不会显示在锁屏界面之上。

第165.2节：在Android 6.0及以上版本授予SYSTEM_ALERT_WINDOW权限

从Android 6.0开始，该权限需要动态授予，

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```

在6.0上抛出以下权限拒绝错误，

原因： android.view.WindowManager\$BadTokenException：无法添加窗口
 android.view.ViewRootImpl\$W@86fb55b -- 此窗口类型权限被拒绝

解决方案：

Chapter 165: Creating Overlay (always-on-top) Windows

Section 165.1: Popup overlay

In order to put your view on top of every application, you have to assign your view to the corresponding window manager. For that you need the system alert permission, which can be requested by adding the following line to your manifest file:

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

Note: If your application gets destroyed, your view will be removed from the window manager. Therefore, it is better to create the view and assign it to the window manager by a foreground service.

Assigning a view to the WindowManager

You can retrieve a window manager instance as follows:

```
WindowManager mWindowManager = (WindowManager) mContext.getSystemService(Context.WINDOW_SERVICE);
```

In order to define the position of your view, you have to create some layout parameters as follows:

```
WindowManager.LayoutParams mLayoutParams = new WindowManager.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_PHONE,
    WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON,
    PixelFormat.TRANSLUCENT);
mLayoutParams.gravity = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;
```

Now, you can assign your view together with the created layout parameters to the window manager instance as follows:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voila! Your view has been successfully placed on top of all other applications.

Note: Your view will not be put on top of the keyguard.

Section 165.2: Granting SYSTEM_ALERT_WINDOW Permission on android 6.0 and above

From android 6.0 this permission needs to grant dynamically,

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```

Throwing below permission denied error on 6.0,

Caused by: android.view.WindowManager\$BadTokenException: Unable to add window
 android.view.ViewRootImpl\$W@86fb55b -- permission denied for this window type

Solution:

按如下方式请求覆盖权限，

```
if(!Settings.canDrawOverlays(this)){
    // 请求设置
Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
    Uri.parse("package:" + getPackageName()));
startActivityForResult(intent, REQUEST_OVERLAY_PERMISSION);
}
```

检查结果，

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_OVERLAY_PERMISSION) {
        if (Settings.canDrawOverlays(this)) {
            // 权限已授予...
        } else{
            // 权限未授予...
        }
    }
}
```

Requesting Overlay permission as below,

```
if(!Settings.canDrawOverlays(this)){
    // ask for setting
Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
    Uri.parse("package:" + getPackageName()));
startActivityForResult(intent, REQUEST_OVERLAY_PERMISSION);
}
```

Check for the result,

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_OVERLAY_PERMISSION) {
        if (Settings.canDrawOverlays(this)) {
            // permission granted...
        }else{
            // permission not granted...
        }
    }
}
```

第166章：ExoPlayer

第166.1节：将ExoPlayer添加到项目中

通过jCenter

在项目的 build.gradle 文件中包含以下内容：

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

其中 rX.X.X 是您选择的版本。有关最新版本，请参见项目的 Releases。更多详情，请参见项目在 Bintray 上的信息。

第166.2节：使用 ExoPlayer

实例化您的 ExoPlayer：

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

如果只播放音频，可以使用以下值：

```
RENDERER_COUNT = 1 //因为您只想渲染简单音频  
minBufferMs = 1000  
minRebufferMs = 5000
```

这两个缓冲值可以根据您的需求进行调整。

现在你需要创建一个数据源。当你想要流式传输mp3时，可以使用DefaultUriDataSource。你需要传入上下文和一个用户代理。为了简单起见，播放本地文件时将userAgent传入null：

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

然后创建 sampleSource：

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(  
    uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri 指向你的文件，作为 Extractor，如果你想播放 mp3，可以使用简单的默认 Mp3Extractor。
requestedBufferSize 可以根据你的需求再次调整。例如使用 5000。

现在你可以使用 sample source 创建音频轨道渲染器，方法如下：

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

最后调用 exoPlayer 实例的 prepare 方法：

```
exoPlayer.prepare(audioRenderer);
```

要开始播放，调用：

```
exoPlayer.setPlayWhenReady(true);
```

Chapter 166: ExoPlayer

Section 166.1: Add ExoPlayer to the project

Via jCenter

including the following in your project's build.gradle file:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

where rX.X.X is the your preferred version. For the latest version, see the project's [Releases](#). For more details, see the project on [Bintray](#).

Section 166.2: Using ExoPlayer

Instantiate your ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

To play audio only you can use these values:

```
RENDERER_COUNT = 1 //since you want to render simple audio  
minBufferMs = 1000  
minRebufferMs = 5000
```

Both buffer values can be tweaked according to your requirements.

Now you have to create a DataSource. When you want to stream mp3 you can use the DefaultUriDataSource. You have to pass the Context and a UserAgent. To keep it simple play a local file and pass null as userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Then create the sampleSource:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(  
    uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri points to your file, as an Extractor you can use a simple default Mp3Extractor if you want to play mp3.
requestedBufferSize can be tweaked again according to your requirements. Use 5000 for example.

Now you can create your audio track renderer using the sample source as follows:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Finally call prepare on your exoPlayer instance:

```
exoPlayer.prepare(audioRenderer);
```

To start playback call:

```
exoPlayer.setPlayWhenReady(true);
```

第166.3节：使用标准TrackRenderer实现播放视频和音频的主要步骤

```
// 1. 实例化播放器。  
player = ExoPlayer.Factory.newInstance(RENDERER_COUNT);  
// 2. 构建渲染器。  
MediaCodecVideoTrackRenderer videoRenderer = ...  
MediaCodecAudioTrackRenderer audioRenderer = ...  
// 3. 通过prepare注入渲染器。  
player.prepare(videoRenderer, audioRenderer);  
// 4. 将表面传递给视频渲染器。  
player.sendMessage(videoRenderer, MediaCodecVideoTrackRenderer.MSG_SET_SURFACE, surface);  
// 5. 开始播放。  
player.setPlayWhenReady(true);  
...  
player.release(); // 完成后别忘了释放资源！
```

Section 166.3: Main steps to play video & audio using the standard TrackRenderer implementations

```
// 1. Instantiate the player.  
player = ExoPlayer.Factory.newInstance(RENDERER_COUNT);  
// 2. Construct renderers.  
MediaCodecVideoTrackRenderer videoRenderer = ...  
MediaCodecAudioTrackRenderer audioRenderer = ...  
// 3. Inject the renderers through prepare.  
player.prepare(videoRenderer, audioRenderer);  
// 4. Pass the surface to the video renderer.  
player.sendMessage(videoRenderer, MediaCodecVideoTrackRenderer.MSG_SET_SURFACE, surface);  
// 5. Start playback.  
player.setPlayWhenReady(true);  
...  
player.release(); // Don't forget to release when done!
```

第167章：XMPP注册登录和聊天 简单示例

第167.1节：XMPP注册登录和聊天基础示例

在你的系统或服务器上安装openfire或任何聊天服务器。更多详情点击[这里](#)。

创建Android项目并在gradle中添加以下库：

```
compile 'org.igniterealtime.smack:smack-android:4.2.0'
compile 'org.igniterealtime.smack:smack-tcp:4.2.0'
compile 'org.igniterealtime.smack:smack-im:4.2.0'
compile 'org.igniterealtime.smack:smack-android-extensions:4.2.0'
```

接下来创建一个用于XMPP连接的XMPP类：

```
public class XMPP {

    public static final int PORT = 5222;
    private static XMPP instance;
    private XMPPTCConnection connection;
    private static String TAG = "XMPP-EXAMPLE";
    public static final String ACTION_LOGGED_IN = "liveapp.loggedin";
    private String HOST = "192.168.0.10";

    private XMPPTCConnectionConfiguration buildConfiguration() throws XmppStringprepException {
        XMPPTCConnectionConfiguration.Builder builder =
            XMPPTCConnectionConfiguration.builder();

        builder.setHost(HOST);
        builder.setPort(PORT);
        builder.setCompressionEnabled(false);
        builder.setDebuggerEnabled(true);
        builder.setSecurityMode(ConnectionConfiguration.SecurityMode.disabled);
        builder.setSendPresence(true);

        if (Build.VERSION.SDK_INT >= 14) {
            builder.setKeystoreType("AndroidCAStore");
            // config.setTruststorePassword(null);
        }
        builder.setKeystorePath(null);
    } else {
        builder.setKeystoreType("BKS");
        String str = System.getProperty("javax.net.ssl.trustStore");
        if (str == null) {
            str = System.getProperty("java.home") + File.separator + "etc" + File.separator +
                "security"
                + File.separator + "cacerts.bks";
        }
        builder.setKeystorePath(str);
    }

    DomainBareJid serviceName = JidCreate.domainBareFrom(HOST);
    builder.setServiceName(serviceName);

    return builder.build();
}
```

Chapter 167: XMPP register login and chat simple example

Section 167.1: XMPP register login and chat basic example

Install openfire or any chat server in your system or on server. For more details [click here](#).

Create android project and add these libraries in gradle:

```
compile 'org.igniterealtime.smack:smack-android:4.2.0'
compile 'org.igniterealtime.smack:smack-tcp:4.2.0'
compile 'org.igniterealtime.smack:smack-im:4.2.0'
compile 'org.igniterealtime.smack:smack-android-extensions:4.2.0'
```

Next create one xmpp class from xmpp connection purpose:

```
public class XMPP {

    public static final int PORT = 5222;
    private static XMPP instance;
    private XMPPTCConnection connection;
    private static String TAG = "XMPP-EXAMPLE";
    public static final String ACTION_LOGGED_IN = "liveapp.loggedin";
    private String HOST = "192.168.0.10";

    private XMPPTCConnectionConfiguration buildConfiguration() throws XmppStringprepException {
        XMPPTCConnectionConfiguration.Builder builder =
            XMPPTCConnectionConfiguration.builder();

        builder.setHost(HOST);
        builder.setPort(PORT);
        builder.setCompressionEnabled(false);
        builder.setDebuggerEnabled(true);
        builder.setSecurityMode(ConnectionConfiguration.SecurityMode.disabled);
        builder.setSendPresence(true);

        if (Build.VERSION.SDK_INT >= 14) {
            builder.setKeystoreType("AndroidCAStore");
            // config.setTruststorePassword(null);
            builder.setKeystorePath(null);
        } else {
            builder.setKeystoreType("BKS");
            String str = System.getProperty("javax.net.ssl.trustStore");
            if (str == null) {
                str = System.getProperty("java.home") + File.separator + "etc" + File.separator +
                    "security"
                    + File.separator + "cacerts.bks";
            }
            builder.setKeystorePath(str);
        }

        DomainBareJid serviceName = JidCreate.domainBareFrom(HOST);
        builder.setServiceName(serviceName);

        return builder.build();
    }
}
```

```

private XMPPTCPConnection getConnection() throws XMPPException, SmackException, IOException,
InterruptedException {
    Log.logDebug(TAG, "获取 XMPP 连接");
    if (已连接()) {
        Log.logDebug(TAG, "返回已存在的连接");
        return this.connection;
    }

    long l = System.currentTimeMillis();
    try {
        if(this.connection != null){
            Log.logDebug(TAG, "找到连接, 尝试连接");
            this.connection.connect();
        } else{
            Log.logDebug(TAG, "未找到连接, 尝试创建新连接");
            XMPPTCPConnectionConfiguration config = buildConfiguration();
            SmackConfiguration.DEBUG = true;
            this.connection = new XMPPTCPConnection(config);
            this.connection.connect();
        }
    } 捕获 (异常 e) {
        Log.LogError(TAG,"获取连接时出现问题：" + e.getMessage());
    }

    Log.logDebug(TAG, "连接属性: " + connection.getHost() + " " +
connection.getServiceName());
    Log.logDebug(TAG, "首次连接耗时: " + (System.currentTimeMillis() - l));
    return this.connection;
}

public static XMPP getInstance() {
    if (instance == null) {
        synchronized (XMPP.class) {
            if (instance == null) {
                instance = new XMPP();
            }
        }
    }
    return instance;
}

public void close() {
    Log.logInfo(TAG, "进入 XMPP 关闭方法");
    if (this.connection != null) {
        this.connection.disconnect();
    }
}

private XMPPTCPConnection connectAndLogin(Context context) {
    Log.logDebug(TAG, "进入连接和登录方法");
    if (!isConnected()) {
        Log.logDebug(TAG, "连接未连接, 尝试登录并连接");
        try {
            // 保存用户名和密码然后在这里使用
            String username = AppSettings.getUser(context);
            String password = AppSettings.getPassword(context);
            this.connection = getConnection();
            Log.logDebug(TAG, "XMPP 用户名 :" + username);
            Log.logDebug(TAG, "XMPP 密码 :" + password);
            this.connection.login(username, password);
            Log.logDebug(TAG, "连接和登录方法, 登录成功");
        }
    }
}

```

```

private XMPPTCPConnection getConnection() throws XMPPException, SmackException, IOException,
InterruptedException {
    Log.logDebug(TAG, "Getting XMPP Connect");
    if (isConnected()) {
        Log.logDebug(TAG, "Returning already existing connection");
        return this.connection;
    }

    long l = System.currentTimeMillis();
    try {
        if(this.connection != null){
            Log.logDebug(TAG, "Connection found, trying to connect");
            this.connection.connect();
        } else{
            Log.logDebug(TAG, "No Connection found, trying to create a new connection");
            XMPPTCPConnectionConfiguration config = buildConfiguration();
            SmackConfiguration.DEBUG = true;
            this.connection = new XMPPTCPConnection(config);
            this.connection.connect();
        }
    } catch (Exception e) {
        Log.LogError(TAG,"some issue with getting connection：" + e.getMessage());
    }

    Log.logDebug(TAG, "Connection Properties: " + connection.getHost() + " " +
connection.getServiceName());
    Log.logDebug(TAG, "Time taken in first time connect: " + (System.currentTimeMillis() - l));
    return this.connection;
}

public static XMPP getInstance() {
    if (instance == null) {
        synchronized (XMPP.class) {
            if (instance == null) {
                instance = new XMPP();
            }
        }
    }
    return instance;
}

public void close() {
    Log.logInfo(TAG, "Inside XMPP close method");
    if (this.connection != null) {
        this.connection.disconnect();
    }
}

private XMPPTCPConnection connectAndLogin(Context context) {
    Log.logDebug(TAG, "Inside connect and Login");
    if (!isConnected()) {
        Log.logDebug(TAG, "Connection not connected, trying to login and connect");
        try {
            // Save username and password then use here
            String username = AppSettings.getUser(context);
            String password = AppSettings.getPassword(context);
            this.connection = getConnection();
            Log.logDebug(TAG, "XMPP username :" + username);
            Log.logDebug(TAG, "XMPP password :" + password);
            this.connection.login(username, password);
            Log.logDebug(TAG, "Connect and Login method, Login successful");
        }
    }
}

```

```

context.sendBroadcast(new Intent(ACTION_LOGGED_IN));
    } catch (XMPPException localXMPPException) {
Log.LogError(TAG, "连接和登录方法出错");
    localXMPPException.printStackTrace();
} catch (SmackException e) {
Log.LogError(TAG, "连接和登录方法出错");
e.printStackTrace();
} 捕获 (IOException e) {
Log.LogError(TAG, "连接和登录方法出错");
e.printStackTrace();
} 捕获 (InterruptedException e) {
    Log.LogError(TAG, "连接和登录方法中的错误");
e.printStackTrace();
} 捕获 (IllegalArgumentException e) {
    Log.LogError(TAG, "连接和登录方法中的错误");
e.printStackTrace();
} 捕获 (异常 e) {
Log.LogError(TAG, "连接和登录方法出错");
e.printStackTrace();
}
}

Log.logInfo(TAG, "进入 getConnection - 返回连接");
return this.connection;
}

public boolean isConnected() {
    return (this.connection != null) && (this.connection.isConnected());
}

public EntityFullJid getUser() {
    if (已连接()) {
        return connection.getUser();
    } else {
        return null;
    }
}

public void login(String user, String pass, String username)
    throws XMPPException, SmackException, IOException, InterruptedException,
PurplKiteXMPPConnectException {
Log.logInfo(TAG, "inside XMPP getlogin Method");
    long l = System.currentTimeMillis();
    XMPPTCPConnection connect = getConnection();
    if (connect.isAuthenticated()) {
        Log.logInfo(TAG, "User already logged in");
        return;
    }

Log.logInfo(TAG, "Time taken to connect: " + (System.currentTimeMillis() - l));

    l = System.currentTimeMillis();
    try{
connect.login(user, pass);
    }catch (Exception e){
Log.LogError(TAG, "Issue in login, check the stacktrace");
e.printStackTrace();
}

Log.logInfo(TAG, "登录耗时: " + (System.currentTimeMillis() - l));

    Log.logInfo(TAG, "登录步骤通过");
}

```

```

context.sendBroadcast(new Intent(ACTION_LOGGED_IN));
    } catch (XMPPException localXMPPException) {
Log.LogError(TAG, "Error in Connect and Login Method");
    localXMPPException.printStackTrace();
} catch (SmackException e) {
    Log.LogError(TAG, "Error in Connect and Login Method");
    e.printStackTrace();
} catch (IOException e) {
    Log.LogError(TAG, "Error in Connect and Login Method");
    e.printStackTrace();
} catch (InterruptedException e) {
    Log.LogError(TAG, "Error in Connect and Login Method");
    e.printStackTrace();
} catch (IllegalArgumentException e) {
    Log.LogError(TAG, "Error in Connect and Login Method");
    e.printStackTrace();
} catch (Exception e) {
    Log.LogError(TAG, "Error in Connect and Login Method");
    e.printStackTrace();
}

Log.logInfo(TAG, "Inside getConnection - Returning connection");
return this.connection;
}

public boolean isConnected() {
    return (this.connection != null) && (this.connection.isConnected());
}

public EntityFullJid getUser() {
    if (isConnected()) {
        return connection.getUser();
    } else {
        return null;
    }
}

public void login(String user, String pass, String username)
    throws XMPPException, SmackException, IOException, InterruptedException,
PurplKiteXMPPConnectException {
Log.logInfo(TAG, "inside XMPP getlogin Method");
    long l = System.currentTimeMillis();
    XMPPTCPConnection connect = getConnection();
    if (connect.isAuthenticated()) {
        Log.logInfo(TAG, "User already logged in");
        return;
    }

Log.logInfo(TAG, "Time taken to connect: " + (System.currentTimeMillis() - l));

    l = System.currentTimeMillis();
    try{
connect.login(user, pass);
    }catch (Exception e){
        Log.LogError(TAG, "Issue in login, check the stacktrace");
        e.printStackTrace();
    }

Log.logInfo(TAG, "Time taken to login: " + (System.currentTimeMillis() - l));

    Log.logInfo(TAG, "login step passed");
}

```

```

PingManager pingManager = PingManager.getInstanceFor(connect);
pingManager.setPingInterval(5000);

}

public void register(String user, String pass) throws XMPPException,
SmackException.NoResponseException, SmackException.NotConnectedException {
    Log.logInfo(TAG, "进入XMPP注册方法, " + user + ":" + pass);
    long l = System.currentTimeMillis();
    try {
        AccountManager accountManager = AccountManager.getInstance(getConnection());
        accountManager.sensitiveOperationOverInsecureConnection(true);
        accountManager.createAccount(Localpart.from(user), pass);
    } catch (SmackException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (PurplKiteXMPPConnectException e) {
        e.printStackTrace();
    }
    Log.logInfo(TAG, "注册耗时: " + (System.currentTimeMillis() - l));
}

public void addStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection = connectAndLogin(context);
    connection.addAsyncStanzaListener(stanzaListener, null);
}

public void removeStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection = connectAndLogin(context);
    connection.removeAsyncStanzaListener(stanzaListener);
}

public void addChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context))
    .addChatListener(chatManagerListener);
}

public void removeChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context)).removeChatListener(chatManagerListener);
}

public void getSrvDeliveryManager(Context context){
    ServiceDiscoveryManager sdm = ServiceDiscoveryManager
        .getInstanceFor(XMPP.getInstance().connectAndLogin(
            context));
    //sdm.addFeature("http://jabber.org/protocol/disco#info");
    //sdm.addFeature("jabber:iq:privacy");
    sdm.addFeature("jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/disco#info");
    sdm.addFeature("jabber:iq:privacy");
}

public String getUserLocalPart(Context context){
    return connectAndLogin(context).getUser().getLocalpart().toString();
}

```

```

PingManager pingManager = PingManager.getInstanceFor(connect);
pingManager.setPingInterval(5000);

}

public void register(String user, String pass) throws XMPPException,
SmackException.NoResponseException, SmackException.NotConnectedException {
    Log.logInfo(TAG, "inside XMPP register method, " + user + ":" + pass);
    long l = System.currentTimeMillis();
    try {
        AccountManager accountManager = AccountManager.getInstance(getConnection());
        accountManager.sensitiveOperationOverInsecureConnection(true);
        accountManager.createAccount(Localpart.from(user), pass);
    } catch (SmackException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (PurplKiteXMPPConnectException e) {
        e.printStackTrace();
    }
    Log.logInfo(TAG, "Time taken to register: " + (System.currentTimeMillis() - l));
}

public void addStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection = connectAndLogin(context);
    connection.addAsyncStanzaListener(stanzaListener, null);
}

public void removeStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection = connectAndLogin(context);
    connection.removeAsyncStanzaListener(stanzaListener);
}

public void addChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context))
        .addChatListener(chatManagerListener);
}

public void removeChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context)).removeChatListener(chatManagerListener);
}

public void getSrvDeliveryManager(Context context){
    ServiceDiscoveryManager sdm = ServiceDiscoveryManager
        .getInstanceFor(XMPP.getInstance().connectAndLogin(
            context));
    //sdm.addFeature("http://jabber.org/protocol/disco#info");
    //sdm.addFeature("jabber:iq:privacy");
    sdm.addFeature("jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/disco#info");
    sdm.addFeature("jabber:iq:privacy");
}

public String getUserLocalPart(Context context){
    return connectAndLogin(context).getUser().getLocalpart().toString();
}

```

```

public EntityFullJid getUser(Context context){
    return connectAndLogin(context).getUser();
}

public Chat getThreadChat(Context context, String party1, String party2){
    Chat chat = ChatManager.getInstanceFor(
        XMPP.getInstance().connectAndLogin(context))
        .getThreadChat(party1 + " - " + party2);
    return chat;
}

public Chat createChat(Context context, EntityJid jid, String party1, String party2,
    ChatMessageListener messageListener){
    Chat chat = ChatManager.getInstanceFor(
        XMPP.getInstance().connectAndLogin(context))
        .createChat(jid, party1 + " - " + party2,
            messageListener);
    return chat;
}

public void sendPacket(Context context, Stanza packet){
    try {
        connectAndLogin(context).sendStanza(packet);
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

最后，添加此活动：

```

private UserLoginTask mAuthTask = null;
private ChatManagerListener chatListener;
private Chat chat;
private Jid opt_jid;
private ChatMessageListener messageListener;
private StanzaListener packetListener;

private boolean register(final String paramString1,final String paramString2) {
    try {
        XMPP.getInstance().register(paramString1, paramString2);
        return true;
    } catch (XMPPException localXMPPException) {
        localXMPPException.printStackTrace();
    } catch (SmackException.NoResponseException e) {
        e.printStackTrace();
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    }
    return false;
}

private boolean login(final String user,final String pass,final String username) {
    try {
        XMPP.getInstance().login(user, pass, username);
    }

```

```

public EntityFullJid getUser(Context context){
    return connectAndLogin(context).getUser();
}

public Chat getThreadChat(Context context, String party1, String party2){
    Chat chat = ChatManager.getInstanceFor(
        XMPP.getInstance().connectAndLogin(context))
        .getThreadChat(party1 + " - " + party2);
    return chat;
}

public Chat createChat(Context context, EntityJid jid, String party1, String party2,
    ChatMessageListener messageListener){
    Chat chat = ChatManager.getInstanceFor(
        XMPP.getInstance().connectAndLogin(context))
        .createChat(jid, party1 + " - " + party2,
            messageListener);
    return chat;
}

public void sendPacket(Context context, Stanza packet){
    try {
        connectAndLogin(context).sendStanza(packet);
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

Finally, add this activiy:

```

private UserLoginTask mAuthTask = null;
private ChatManagerListener chatListener;
private Chat chat;
private Jid opt_jid;
private ChatMessageListener messageListener;
private StanzaListener packetListener;

private boolean register(final String paramString1,final String paramString2) {
    try {
        XMPP.getInstance().register(paramString1, paramString2);
        return true;
    } catch (XMPPException localXMPPException) {
        localXMPPException.printStackTrace();
    } catch (SmackException.NoResponseException e) {
        e.printStackTrace();
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    }
    return false;
}

private boolean login(final String user,final String pass,final String username) {
    try {
        XMPP.getInstance().login(user, pass, username);
    }

```

```

sendBroadcast(new Intent("liveapp.loggedin"));

    return true;
} catch (Exception e) {
e.printStackTrace();
try {

XMPP.getInstance()
    .login(user, pass, username);
sendBroadcast(new Intent("liveapp.loggedin"));

    return true;
} catch (XMPPException e1) {
e1.printStackTrace();
} catch (SmackException e1) {
e1.printStackTrace();
} catch (InterruptedException e1) {
e1.printStackTrace();
} catch (IOException e1) {
e1.printStackTrace();
} catch (Exception e1){
e1.printStackTrace();
}
}

return false;
}

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

public UserLoginTask() {
}

protected Boolean doInBackground(Void... paramVarArgs) {
String mEmail = "abc";
String mUsername = "abc";
String mPassword = "welcome";

if (register(mEmail, mPassword)) {
try {
XMPP.getInstance().close();
} catch (Exception e) {
e.printStackTrace();
}
}
return login(mEmail, mPassword, mUsername);
}

protected void onCancelled() {
mAuthTask = null;
}

@Override
protected void onPreExecute() {
super.onPreExecute();
}

protected void onPostExecute(Boolean success) {
mAuthTask = null;
try {

```

```

sendBroadcast(new Intent("liveapp.loggedin"));

    return true;
} catch (Exception e) {
e.printStackTrace();
try {

XMPP.getInstance()
    .login(user, pass, username);
sendBroadcast(new Intent("liveapp.loggedin"));

    return true;
} catch (XMPPException e1) {
e1.printStackTrace();
} catch (SmackException e1) {
e1.printStackTrace();
} catch (InterruptedException e1) {
e1.printStackTrace();
} catch (IOException e1) {
e1.printStackTrace();
} catch (Exception e1){
e1.printStackTrace();
}
}

return false;
}

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

public UserLoginTask() {
}

protected Boolean doInBackground(Void... paramVarArgs) {
String mEmail = "abc";
String mUsername = "abc";
String mPassword = "welcome";

if (register(mEmail, mPassword)) {
try {
XMPP.getInstance().close();
} catch (Exception e) {
e.printStackTrace();
}
}
return login(mEmail, mPassword, mUsername);
}

protected void onCancelled() {
mAuthTask = null;
}

@Override
protected void onPreExecute() {
super.onPreExecute();
}

protected void onPostExecute(Boolean success) {
mAuthTask = null;
try {

```

```

if (success) {

messageListener = new ChatMessageListener() {
    @Override
    public void processMessage(Chat chat, Message message) {
        // here you will get only connected user by you
    }
};

packetListener = new StanzaListener() {
    @Override
    public void processPacket(Stanza packet) throws
SmackException.NotConnectedException, InterruptedException {
        if (packet instanceof Message) {
            final Message message = (Message) packet;
            // 这里你将收到任何人发送的所有消息
        }
    }
};

chatListener = new ChatManagerListener() {
    @Override
    public void chatCreated(Chat chatCreated, boolean local) {
        onChatCreated(chatCreated);
    }
};

try {
    String opt_jidStr = "abc";

    try {
        opt_jid = JidCreate.bareFrom(Localpart.from(opt_jidStr), Domainpart.from(HOST));
    } catch (XmppStringprepException e) {
        e.printStackTrace();
    }
    String addr1 = XMPP.getInstance().getUserLocalPart(getActivity());
    String addr2 = opt_jid.toString();
    if (addr1.compareTo(addr2) > 0) {
        addr2 = addr1;
        addr1 = addr3;
    }
    chat = XMPP.getInstance().getThreadChat(getActivity(), addr1, addr2);
    if (chat == null) {
        chat = XMPP.getInstance().createChat(getActivity(), (EntityJid) opt_jid, addr1, addr2,
messageListener);
        PurplkiteLogs.logInfo(TAG, "chat value single chat 1 :" + chat);
    } else {
        chat.addMessageListener(messageListener);
        PurplkiteLogs.logInfo(TAG, "chat value single chat 2:" + chat);
    }
} 捕获 (异常 e) {
e.printStackTrace();
}

```

```

if (success) {

messageListener = new ChatMessageListener() {
    @Override
    public void processMessage(Chat chat, Message message) {
        // here you will get only connected user by you
    }
};

packetListener = new StanzaListener() {
    @Override
    public void processPacket(Stanza packet) throws
SmackException.NotConnectedException, InterruptedException {
        if (packet instanceof Message) {
            final Message message = (Message) packet;
            // here you will get all messages send by anybody
        }
    }
};

chatListener = new ChatManagerListener() {
    @Override
    public void chatCreated(Chat chatCreated, boolean local) {
        onChatCreated(chatCreated);
    }
};

try {
    String opt_jidStr = "abc";

    try {
        opt_jid = JidCreate.bareFrom(Localpart.from(opt_jidStr), Domainpart.from(HOST));
    } catch (XmppStringprepException e) {
        e.printStackTrace();
    }
    String addr1 = XMPP.getInstance().getUserLocalPart(getActivity());
    String addr2 = opt_jid.toString();
    if (addr1.compareTo(addr2) > 0) {
        String addr3 = addr2;
        addr2 = addr1;
        addr1 = addr3;
    }
    chat = XMPP.getInstance().getThreadChat(getActivity(), addr1, addr2);
    if (chat == null) {
        chat = XMPP.getInstance().createChat(getActivity(), (EntityJid) opt_jid, addr1, addr2,
messageListener);
        PurplkiteLogs.logInfo(TAG, "chat value single chat 1 :" + chat);
    } else {
        chat.addMessageListener(messageListener);
        PurplkiteLogs.logInfo(TAG, "chat value single chat 2:" + chat);
    }
} catch (Exception e) {
e.printStackTrace();
}

```

```

XMPP.getInstance().addStanzaListener(getActivity(), packetListener);
XMPP.getInstance().addChatListener(getActivity(), chatListener);
XMPP.getInstance().getSrvDeliveryManager(getActivity());

} else {

}

} 捕获 (异常 e) {
e.printStackTrace();
}

}

/***
* 用户尝试登录 xmpp
*/
private void 尝试登录() {
if (mAuthTask != null) {
return;
}

boolean 取消 = false;
View 焦点视图 = null;

if (取消) {
焦点视图.请求焦点();
} else {
try {
mAuthTask = new 用户登录任务();
mAuthTask.执行((Void) null);
} catch (Exception e) {

}
}

void 在聊天创建时(Chat chatCreated) {
if (chat != null) {
if (chat.获取参与者().获取本地部分().toString().等于(
chatCreated.获取参与者().获取本地部分().toString())) {
chat.移除消息监听器(messageListener);
chat = chatCreated;
chat.添加消息监听器(messageListener);
}
} else {
chat = chatCreated;
chat.addMessageListener(messageListener);
}
}

private void sendMessage(String message) {
if (chat != null) {
try {
chat.sendMessage(message);
} catch (SmackException.NotConnectedException e) {
e.printStackTrace();
} catch (Exception e) {
}
}
}

```

```

XMPP.getInstance().addStanzaListener(getActivity(), packetListener);
XMPP.getInstance().addChatListener(getActivity(), chatListener);
XMPP.getInstance().getSrvDeliveryManager(getActivity());

} else {

}

} catch (Exception e) {
e.printStackTrace();
}

}

/***
* user attemptLogin for xmpp
*/
private void attemptLogin() {
if (mAuthTask != null) {
return;
}

boolean cancel = false;
View focusView = null;

if (cancel) {
focusView.requestFocus();
} else {
try {
mAuthTask = new UserLoginTask();
mAuthTask.execute((Void) null);
} catch (Exception e) {

}
}

void onChatCreated(Chat chatCreated) {
if (chat != null) {
if (chat.getParticipant().getLocalpart().toString().equals(
chatCreated.getParticipant().getLocalpart().toString())) {
chat.removeMessageListener(messageListener);
chat = chatCreated;
chat.addMessageListener(messageListener);
}
} else {
chat = chatCreated;
chat.addMessageListener(messageListener);
}
}

private void sendMessage(String message) {
if (chat != null) {
try {
chat.sendMessage(message);
} catch (SmackException.NotConnectedException e) {
e.printStackTrace();
} catch (Exception e) {
}
}
}

```

```
e.printStackTrace();
    }
}

@Override
public void onDestroy() {
    // TODO Auto-generated method stub
    super.onDestroy();
    try {
XMPP.getInstance().removeChatListener(getActivity(), chatListener);
        if (chat != null && messageListener != null) {
            XMPP.getInstance().removeStanzaListener(getActivity(), packetListener);
            chat.removeMessageListener(messageListener);
        }
    } 捕获 (异常 e) {
e.printStackTrace();
    }
}
```

确保在您的清单文件中添加了互联网权限。

```
e.printStackTrace();
    }
}

@Override
public void onDestroy() {
    // TODO Auto-generated method stub
    super.onDestroy();
    try {
XMPP.getInstance().removeChatListener(getActivity(), chatListener);
        if (chat != null && messageListener != null) {
            XMPP.getInstance().removeStanzaListener(getActivity(), packetListener);
            chat.removeMessageListener(messageListener);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Make sure the internet permission is added in your manifest file.

第168章：Android 认证器

第168.1节：基本账户认证服务

Android账户认证系统可用于使客户端与远程服务器进行身份验证。需要三条信息：

- 一个由`android.accounts.AccountAuthenticator`触发的服务。其`onBind`方法应返回`AbstractAccountAuthenticator`的子类。
- 一个用于提示用户输入凭据的活动（登录活动）
- 一个描述账户的xml资源文件

1. 服务：

在你的`AndroidManifest.xml`中添加以下权限：

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

在清单文件中声明该服务：

```
<service android:name="com.example.MyAuthenticationService">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

请注意，`android.accounts.AccountAuthenticator`包含在`intent-filter`标签内。`xml`资源（此处命名为`authenticator`）在`meta-data`标签中指定。

服务类：

```
public class MyAuthenticationService extends Service {

    private static final Object lock = new Object();
    private MyAuthenticator mAuthenticator;

    public MyAuthenticationService() {
        super();
    }

    @Override
    public void onCreate() {
        super.onCreate();

        synchronized (lock) {
            if (mAuthenticator == null) {
                mAuthenticator = new MyAuthenticator(this);
            }
        }
    }
}
```

Chapter 168: Android Authenticator

Section 168.1: Basic Account Authenticator Service

The Android Account Authenticator system can be used to make the client authenticate with a remote server. Three pieces of information are required:

- A service, triggered by the `android.accounts.AccountAuthenticator`. Its `onBind` method should return a subclass of `AbstractAccountAuthenticator`.
- An activity to prompt the user for credentials (Login activity)
- An xml resource file to describe the account

1. The service:

Place the following permissions in your `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

Declare the service in the manifest file:

```
<service android:name="com.example.MyAuthenticationService">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

Note that the `android.accounts.AccountAuthenticator` is included within the `intent-filter` tag. The `xml` resource (named `authenticator` here) is specified in the `meta-data` tag.

The service class:

```
public class MyAuthenticationService extends Service {

    private static final Object lock = new Object();
    private MyAuthenticator mAuthenticator;

    public MyAuthenticationService() {
        super();
    }

    @Override
    public void onCreate() {
        super.onCreate();

        synchronized (lock) {
            if (mAuthenticator == null) {
                mAuthenticator = new MyAuthenticator(this);
            }
        }
    }
}
```

```

@Override
public IBinder onBind(Intent intent) {
    return mAuthenticator.getIBinder();
}

}

```

2. XML 资源：

```

<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.example.account"
    android:icon="@drawable/appicon"
    android:smallIcon="@drawable/appicon"
    android:label="@string/app_name" />

```

不要直接将字符串赋值给 android:label 或赋值缺失的 drawable。否则会无警告地崩溃。

3. 继承 AbstractAccountAuthenticator 类：

```

public class MyAuthenticator extends AbstractAccountAuthenticator {

    private Context mContext;

    public MyAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response,
                            String accountType,
                            String authTokenType,
                            String[] requiredFeatures,
                            Bundle options) throws NetworkErrorException {

        Intent intent = new Intent(mContext, LoginActivity.class);
        intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE, response);

        Bundle bundle = new Bundle();
        bundle.putParcelable(AccountManager.KEY_INTENT, intent);

        return bundle;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle options) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
}

```

```

@Override
public IBinder onBind(Intent intent) {
    return mAuthenticator.getIBinder();
}

}

```

2. The xml resource:

```

<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.example.account"
    android:icon="@drawable/appicon"
    android:smallIcon="@drawable/appicon"
    android:label="@string/app_name" />

```

Do not directly assign a string to android:label or assign missing drawables. It will crash without warning.

3. Extend the AbstractAccountAuthenticator class:

```

public class MyAuthenticator extends AbstractAccountAuthenticator {

    private Context mContext;

    public MyAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response,
                            String accountType,
                            String authTokenType,
                            String[] requiredFeatures,
                            Bundle options) throws NetworkErrorException {

        Intent intent = new Intent(mContext, LoginActivity.class);
        intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE, response);

        Bundle bundle = new Bundle();
        bundle.putParcelable(AccountManager.KEY_INTENT, intent);

        return bundle;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle options) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
}

```

```

@Override
public String getAuthTokenLabel(String authTokenType) {
    return null;
}

@Override
public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features) throws NetworkErrorException {
    return null;
}

@Override
public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
    return null;
}

```

AbstractAccountAuthenticator类中的addAccount()方法非常重要，因为当从设置中的“添加账户”界面添加账户时会调用此方法。

AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE非常重要，因为它将包含AccountAuthenticatorResponse对象，该对象在用户验证成功后需要返回账户密钥。

```

@Override
public String getAuthTokenLabel(String authTokenType) {
    return null;
}

@Override
public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features) throws NetworkErrorException {
    return null;
}

@Override
public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
    return null;
}

```

The addAccount() method in AbstractAccountAuthenticator class is important as this method is called when adding an account from the "Add Account" screen in under settings.

AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE is important, as it will include the AccountAuthenticatorResponse object that is needed to return the account keys upon successful user verification.

第169章：音频管理器

169.1节：请求短暂音频焦点

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // 你现在拥有音频焦点，可以播放声音。
            // 播放完声音后，你需要归还焦点。
        }
        audioManager.abandonAudioFocus(changedListener);
    }
}
```

169.2节：请求音频焦点

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // 你现在拥有音频焦点，可以播放声音。
        } else if (focusChange == AudioManager.AUDIOFOCUS_REQUEST_FAILED) {
            // 处理失败。
        }
    }
}
```

Chapter 169: AudioManager

Section 169.1: Requesting Transient Audio Focus

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
            // When the sound has been played you give the focus back.
            audioManager.abandonAudioFocus(changedListener);
        }
    }
}
```

Section 169.2: Requesting Audio Focus

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
        } else if (focusChange == AudioManager.AUDIOFOCUS_REQUEST_FAILED) {
            // Handle the failure.
        }
    }
}
```

第170章 : AudioTrack

第170.1节 : 生成特定频率的音调

要播放具有特定音调的声音，我们首先必须创建一个正弦波声音。方法如下。

```
final int duration = 10; // 声音持续时间
final int sampleRate = 22050; // 赫兹 (最大频率为7902.13Hz (B8) )
final int numSamples = duration * sampleRate;
final double samples[] = new double[numSamples];
final short buffer[] = new short[numSamples];
for (int i = 0; i < numSamples; ++i)
{
    samples[i] = Math.sin(2 * Math.PI * i / (sampleRate / note[0])); // 正弦波
    buffer[i] = (short) (samples[i] * Short.MAX_VALUE); // 振幅越大音量越大
}
```

现在我们必须配置AudioTrack以根据生成的缓冲区播放。方法如下

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                                         sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                                         AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                                         AudioTrack.MODE_STATIC);
```

写入生成的缓冲区并播放音轨

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

希望这能帮到你 :)

Chapter 170: AudioTrack

Section 170.1: Generate tone of a specific frequency

To play a sound of with a specific tone, we first have to create a sine wave sound. This is done in the following way.

```
final int duration = 10; // duration of sound
final int sampleRate = 22050; // Hz (maximum frequency is 7902.13Hz (B8))
final int numSamples = duration * sampleRate;
final double samples[] = new double[numSamples];
final short buffer[] = new short[numSamples];
for (int i = 0; i < numSamples; ++i)
{
    samples[i] = Math.sin(2 * Math.PI * i / (sampleRate / note[0])); // Sine wave
    buffer[i] = (short) (samples[i] * Short.MAX_VALUE); // Higher amplitude increases volume
}
```

Now we have to configure AudioTrack to play in accordance with the generated buffer . It is done in the following manner

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                                         sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                                         AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                                         AudioTrack.MODE_STATIC);
```

Write the generated buffer and play the track

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Hope this helps :)

第171章：作业调度

第171.1节：基本用法

创建一个新的 JobService

这是通过继承JobService类并实现/重写所需的方法onStartJob()和onStopJob()来完成的。

```
public class MyJobService extends JobService
{
    final String TAG = getClass().getSimpleName();

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        Log.i(TAG, "作业开始");
        // ... your code here ...

        jobFinished(jobParameters, false); // 表示任务完成且不需要重新调度任务
        return false; // 完成：没有更多工作需要执行
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        Log.w(TAG, "任务已停止");
        return false;
    }
}
```

将新的 JobService 添加到你的 AndroidManifest.xml 中

以下步骤是必需的，否则你将无法运行你的任务：

在你的 AndroidManifest.xml 中，声明你的 MyJobService 类为 `<service>` 元素，放置在 `<application> </application>` 标签之间。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyJobService"
            android:permission="android.permission.BIND_JOB_SERVICE" />
    </application>

```

Chapter 171: Job Scheduling

Section 171.1: Basic usage

Create a new JobService

This is done by extending the JobService class and implementing/overriding the required methods `onStartJob()` and `onStopJob()`.

```
public class MyJobService extends JobService
{
    final String TAG = getClass().getSimpleName();

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        Log.i(TAG, "Job started");
        // ... your code here ...

        jobFinished(jobParameters, false); // signal that we're done and don't want to reschedule
        the job
        return false; // finished: no more work to be done
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        Log.w(TAG, "Job stopped");
        return false;
    }
}
```

Add the new JobService to your AndroidManifest.xml

The following step is *mandatory*, otherwise you won't be able to run your job:

Declare your MyJobService class as a new `<service>` element between `<application> </application>` in your `AndroidManifest.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyJobService"
            android:permission="android.permission.BIND_JOB_SERVICE" />
    </application>

```

设置并运行任务

在您实现了新的JobService并将其添加到您的`AndroidManifest.xml`后，您可以继续进行最后的步骤。

- `onButtonClick_startJob()` 准备并运行一个定期任务。除了定期任务之外，`JobInfo.Builder`还允许指定许多其他设置和约束。例如，你可以定义运行任务时需要插入的充电器或network connection。
- `onButtonClick_stopJob()` 取消所有正在运行的任务

```
public class MainActivity extends AppCompatActivity
{
    final String TAG = getClass().getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onButtonClick_startJob(View v) {
        // 从当前上下文获取 JobScheduler 实例
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);

        // MyJobService 提供任务的实现
        ComponentName jobService = new ComponentName(getApplicationContext(), MyJobService.class);

        // 定义任务将以10秒的间隔周期性运行
        JobInfo jobInfo = new JobInfo.Builder(1, jobService).setPeriodic(10 * 1000).build();

        // 调度/启动任务
        int result = jobScheduler.schedule(jobInfo);
        if (result == JobScheduler.RESULT_SUCCESS)
            Log.d(TAG, "成功调度作业: " + result);
        else
            Log.e(TAG, "结果失败: " + result);
    }

    public void onButtonClick_stopJob(View v) {
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
        Log.d(TAG, "正在停止所有作业... ");
        jobScheduler.cancelAll(); // 取消所有可能正在运行的作业
    }
}
```

调用 `onButtonClick_startJob()` 后，作业将大约每隔10秒运行一次，即使应用处于 `paused` 状态（用户按下主页按钮，应用不再可见）。

在 `onButtonClick_stopJob()` 中，不必取消所有正在运行的作业，也可以调用 `jobScheduler.cancel()` 来根据作业ID取消特定作业。

Setup and run the job

After you implemented a new JobService and added it to your `AndroidManifest.xml`, you can continue with the final steps.

- `onButtonClick_startJob()` prepares and runs a periodical job. Besides periodic jobs, `JobInfo.Builder` allows to specify many other settings and constraints. For example you can define that a *plugged in charger* or a *network connection* is required to run the job.
- `onButtonClick_stopJob()` cancels all running jobs

```
public class MainActivity extends AppCompatActivity
{
    final String TAG = getClass().getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onButtonClick_startJob(View v) {
        // get the jobScheduler instance from current context
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);

        // MyJobService provides the implementation for the job
        ComponentName jobService = new ComponentName(getApplicationContext(), MyJobService.class);

        // define that the job will run periodically in intervals of 10 seconds
        JobInfo jobInfo = new JobInfo.Builder(1, jobService).setPeriodic(10 * 1000).build();

        // schedule/start the job
        int result = jobScheduler.schedule(jobInfo);
        if (result == JobScheduler.RESULT_SUCCESS)
            Log.d(TAG, "Successfully scheduled job: " + result);
        else
            Log.e(TAG, "RESULT_FAILURE: " + result);
    }

    public void onButtonClick_stopJob(View v) {
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
        Log.d(TAG, "Stopping all jobs... ");
        jobScheduler.cancelAll(); // cancel all potentially running jobs
    }
}
```

After calling `onButtonClick_startJob()`, the job will approximately run in intervals of 10 seconds, even when the app is in the `paused` state (user pressed home button and app is no longer visible).

Instead of cancelling all running jobs inside `onButtonClick_stopJob()`, you can also call `jobScheduler.cancel()` to cancel a specific job based on its job ID.

第172章：账户与 账户经理

第172.1节：理解自定义账户/认证

以下示例是对关键概念和基本框架设置的高级概述：

1. 从用户收集凭证（通常来自您创建的登录界面）
2. 使用服务器验证凭证（存储自定义认证）
3. 在设备上存储凭证

扩展 `AbstractAccountAuthenticator` (主要用于检索认证并重新认证)

```
public class AccountAuthenticator extends AbstractAccountAuthenticator {  
  
    @Override  
    public Bundle addAccount(AccountAuthenticatorResponse response, String accountType,  
        String authTokenType, String[] requiredFeatures, Bundle options) {  
        // 启动登录活动的意图  
    }  
  
    @Override  
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle  
options) {  
    }  
  
    @Override  
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {  
    }  
  
    @Override  
    public Bundle 获取认证令牌(AccountAuthenticatorResponse 响应, Account 账户, String  
认证令牌类型,  
Bundle 选项) throws NetworkErrorException {  
        //从账户管理器存储  
        //或自定义存储中检索认证令牌，或重新认证旧令牌并返回新令牌  
    }  
  
    @Override  
    public String 获取认证令牌标签(String 认证令牌类型) {  
    }  
  
    @Override  
    public Bundle 是否具有功能(AccountAuthenticatorResponse 响应, Account 账户, String[]  
功能)  
        throws NetworkErrorException {  
        //检查账户是否支持某些功能  
    }  
  
    @Override  
    public Bundle 更新凭据(AccountAuthenticatorResponse 响应, Account 账户, String  
认证令牌类型,  
Bundle 选项){  
        //当用户的会话已过期或需要更新其之前可用的凭据时，这里是执行该操作的函数。  
    }  
}
```

Chapter 172: Accounts and AccountManager

Section 172.1: Understanding custom accounts/authentication

The following example is high level coverage of the key concepts and basic skeletal setup:

1. Collects credentials from the user (Usually from a login screen you've created)
2. Authenticates the credentials with the server (stores custom authentication)
3. Stores the credentials on the device

Extend an `AbstractAccountAuthenticator` (*Primarily used to retrieve authentication & re-authenticate them*)

```
public class AccountAuthenticator extends AbstractAccountAuthenticator {  
  
    @Override  
    public Bundle addAccount(AccountAuthenticatorResponse response, String accountType,  
        String authTokenType, String[] requiredFeatures, Bundle options) {  
        //intent to start the login activity  
    }  
  
    @Override  
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle  
options) {  
    }  
  
    @Override  
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {  
    }  
  
    @Override  
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String  
authTokenType,  
        Bundle options) throws NetworkErrorException {  
        //retrieve authentication tokens from account manager storage or custom storage or re-  
        authenticate old tokens and return new ones  
    }  
  
    @Override  
    public String getAuthTokenLabel(String authTokenType) {  
    }  
  
    @Override  
    public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]  
features)  
        throws NetworkErrorException {  
        //check whether the account supports certain features  
    }  
  
    @Override  
    public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String  
authTokenType,  
        Bundle options) {  
        //when the user's session has expired or requires their previously available credentials to be  
        updated, here is the function to do it.  
    }  
}
```

```
    }
}
```

创建一个服务 (账户管理器框架通过服务接口连接到扩展的 *AbstractAccountAuthenticator*)

```
public class AuthenticatorService extends Service {
    private AccountAuthenticator 认证器;
    @Override
    public void onCreate(){
        认证器 = new AccountAuthenticator(this);
    }
    @Override
    public IBinder onBind(Intent intent) {
        return 认证器.getIBinder();
    }
}
```

认证器 XML 配置 (账户管理框架所需。这是你在 *Android* 设置 -> 账户中看到的内容)

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="rename.with.your.applicationid"
    android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:smallIcon="@drawable/app_icon" />
```

对 *AndroidManifest.xml* 的更改 (将上述所有概念结合起来, 使其可通过 *AccountManager* 编程使用)

```
<application
...
<service
    android:name=".authenticator.AccountAuthenticatorService"
    android:exported="false"
    android:process=":authentication">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator"/>
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator"/>
</service>
</application>
```

下一个示例将包含如何利用此设置。

```
    }
}
```

Create a service (Account Manager framework connects to the extended *AbstractAccountAuthenticator* through the service interface)

```
public class AuthenticatorService extends Service {
    private AccountAuthenticator authenticator;
    @Override
    public void onCreate(){
        authenticator = new AccountAuthenticator(this);
    }
    @Override
    public IBinder onBind(Intent intent) {
        return authenticator.getIBinder();
    }
}
```

Authenticator XML configuration (The account manager framework requires. This is what you'll see inside Settings -> Accounts in Android)

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="rename.with.your.applicationid"
    android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:smallIcon="@drawable/app_icon" />
```

Changes to the *AndroidManifest.xml* (Bring all the above concepts together to make it usable programmatically through the *AccountManager*)

```
<application
...
<service
    android:name=".authenticator.AccountAuthenticatorService"
    android:exported="false"
    android:process=":authentication">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator"/>
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator"/>
</service>
</application>
```

The next example will contain how to make use of this setup.

第173章：将OpenCV集成到Android Studio

第173.1节：说明

已使用 A.S. v1.4.1 进行测试，但也应适用于更新版本。

1. 使用项目向导创建一个新的 Android Studio 项目（菜单：/文件/新建项目）：

- 将其命名为 "cvtest1"
- 设备类型：**API 19, Android 4.4 (KitKat)**
- 空白活动命名为**MainActivity**

你应该有一个存放该项目的cvtest1目录。 (当你打开项目时，Android Studio 的标题栏会显示 cvtest1 的位置)

2. 验证你的应用是否能正确运行。尝试修改类似“Hello World”的文本，以确认构建/测试周期对你来说是正常的。（我使用的是 API 19 设备的模拟器进行测试）。

3. 下载适用于 Android 的 OpenCV v3.1.0 包，并将其解压到某个临时目录中。
(确保下载的是专门针对 Android 的包，而不仅仅是 OpenCV for Java 包。) 我将此目录称为 "unzip-dir"。
在 **unzip-dir** 下，你应该有一个 **sdk/native/libs** 目录，里面有以 **arm...**、**mips...** 和 **x86...** 开头的子目录（分别对应 Android 支持的不同“架构”）。

4. 从 Android Studio 将 OpenCV 作为模块导入到你的项目中：菜单：/文件/新建/导入模块：

- 源目录：**{unzip-dir}/sdk/java**
- 模块名称：Android Studio 会自动填写此字段为**openCVLibrary310**（确切名称可能无关紧要，但我们就用这个）。
- 点击下一步。你会看到一个包含三个复选框的界面，询问关于jar包、库和导入选项的问题。三个都应勾选
◦ 点击完成。

Android Studio 开始导入模块，并显示一个**import-summary.txt**文件，列出了未导入的内容（主要是javadoc文件）及其他信息。

Chapter 173: Integrate OpenCV into Android Studio

Section 173.1: Instructions

Tested with A.S. v1.4.1 but should work with newer versions too.

1. Create a new Android Studio project using the project wizard (Menu:/File/New Project):

- Call it "**cvtest1**"
- Form factor: **API 19, Android 4.4 (KitKat)**
- **Blank Activity** named **MainActivity**

You should have a **cvtest1** directory where this project is stored. (the title bar of Android studio shows you where cvtest1 is when you open the project)

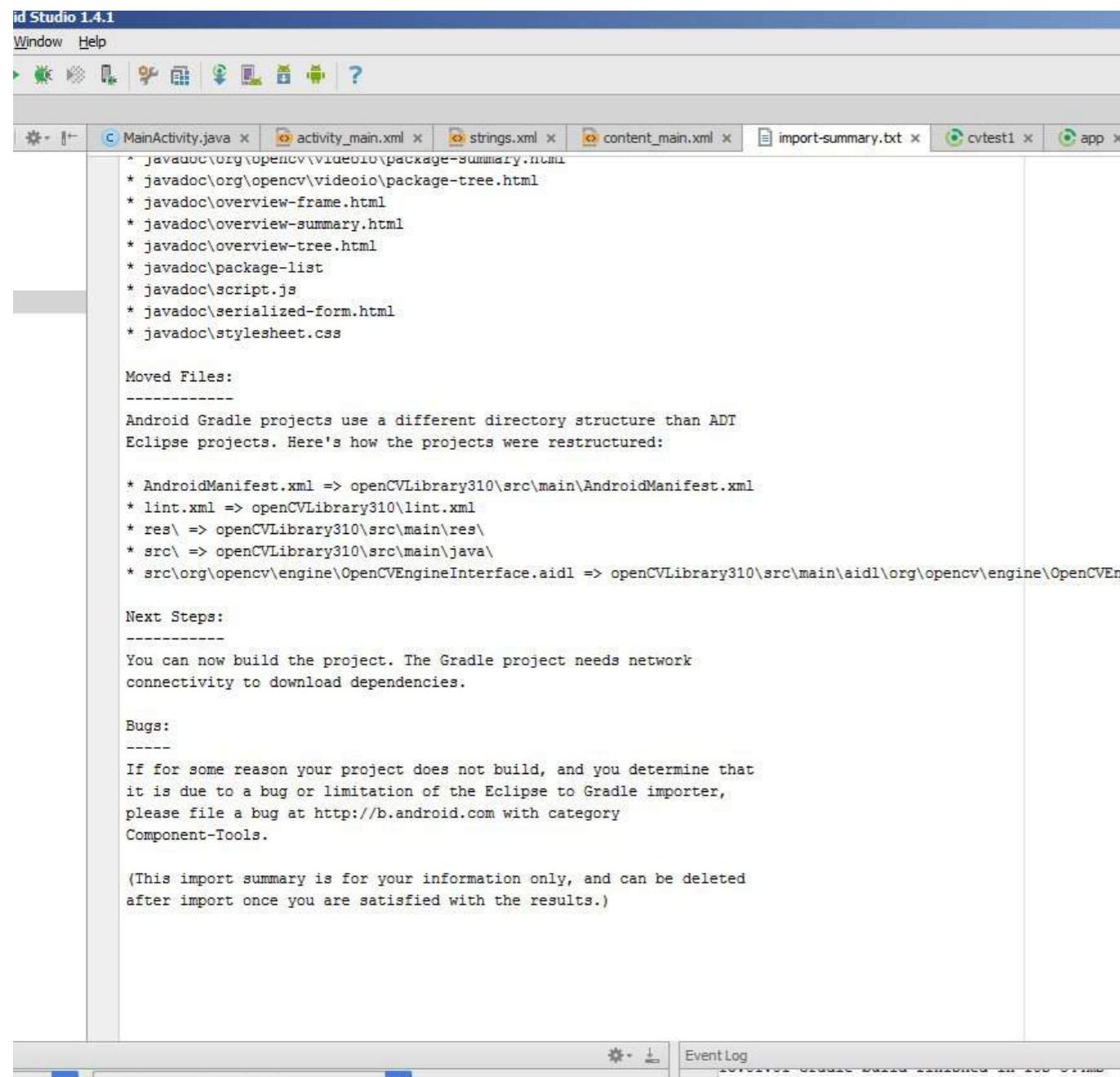
2. Verify that your app runs correctly. Try changing something like the "Hello World" text to confirm that the build/test cycle is OK for you. (I'm testing with an emulator of an API 19 device).

3. Download the OpenCV package for Android v3.1.0 and unzip it in some temporary directory somewhere.
(Make sure it is the package specifically for Android and not just the OpenCV for Java package.) I'll call this directory "**unzip-dir**" Below **unzip-dir** you should have a **sdk/native/libs** directory with subdirectories that start with things like **arm...**, **mips...** and **x86...** (one for each type of "architecture" Android runs on)

4. From Android Studio import OpenCV into your project as a module: **Menu:/File/New/Import_Module:**

- Source-directory: **{unzip-dir}/sdk/java**
- Module name: Android studio automatically fills in this field with **openCVLibrary310** (the exact name probably doesn't matter but we'll go with this).
- Click on **next**. You get a screen with three checkboxes and questions about jars, libraries and import options. All three should be checked. Click on **Finish**.

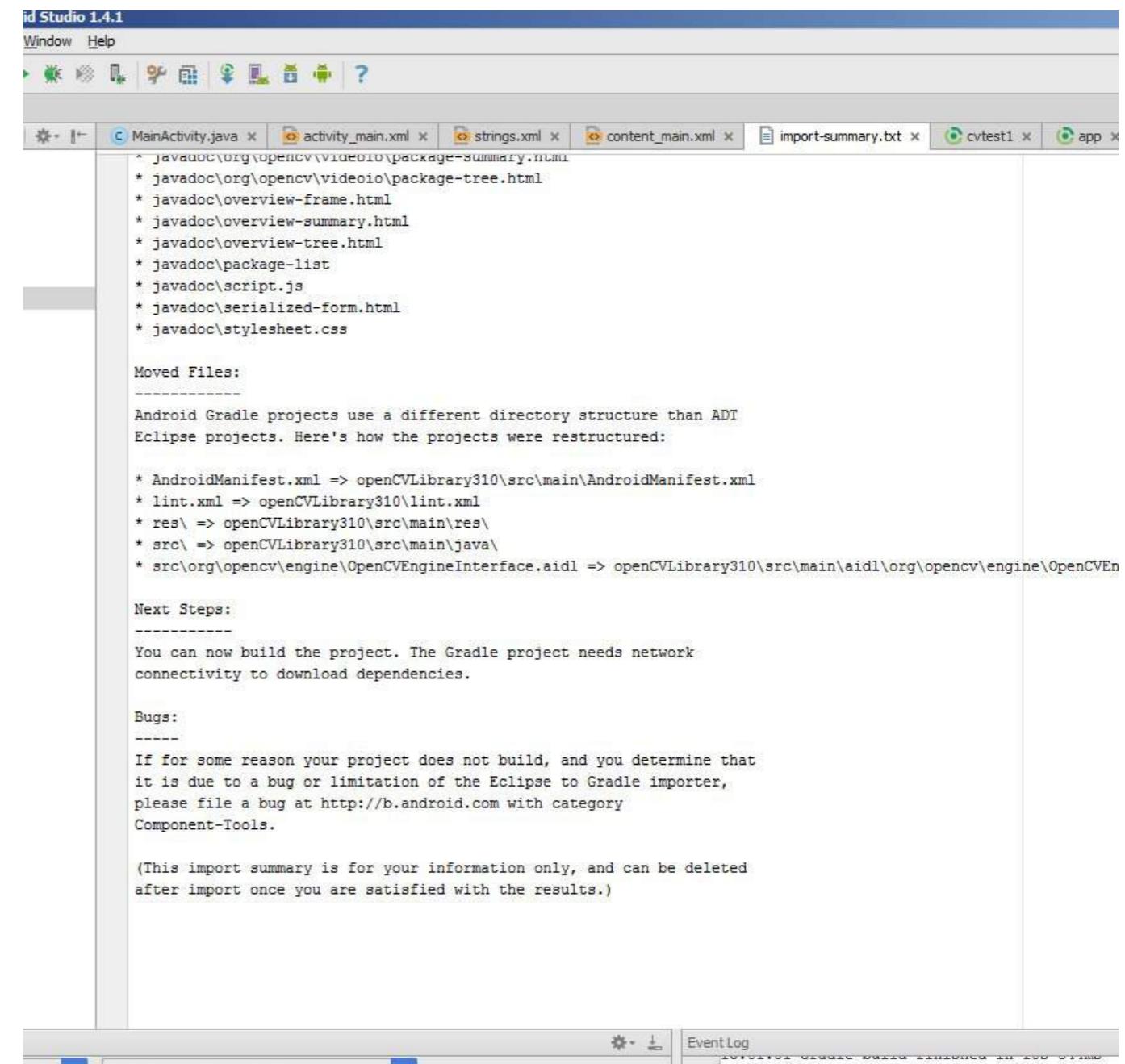
Android Studio starts to import the module and you are shown an **import-summary.txt** file that has a list of what was not imported (mostly javadoc files) and other pieces of information.



但你也会收到一条错误信息，提示未找到哈希字符串为 'android-14' 的目标...。出现此问题是因为你下载的OpenCV压缩包中的build.gradle文件指定使用android API版本14进行编译，而默认情况下你使用的Android Studio v1.4.1并没有该版本。



5. 打开项目结构对话框（菜单：/文件/项目结构）。选择“app”模块，点击
依赖项标签，添加:openCVLibrary310作为模块依赖。当你选择添加/模块依赖时，它应出现在可
添加模块列表中。它现在会显示为依赖项，但事件日志中仍会出现一些找不到android-14的错误。
6. 查看你的app模块的build.gradle文件。Android项目中有多个build.gradle文件。



But you also get an error message saying **failed to find target with hash string 'android-14'**.... This happens because the build.gradle file in the OpenCV zip file you downloaded says to compile using android API version 14, which by default you don't have with Android Studio v1.4.1.

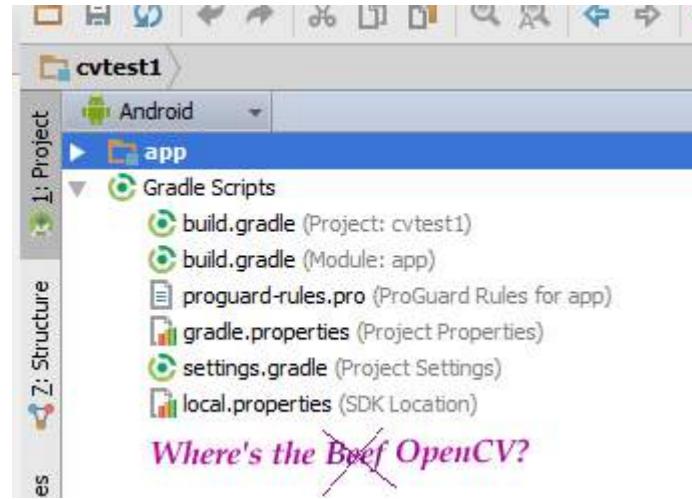


5. Open the project structure dialogue (**Menu:/File/Project_Structure**). Select the "app" module, click on the **Dependencies** tab and add **:openCVLibrary310** as a Module Dependency. When you select **Add/Module_Dependency** it should appear in the list of modules you can add. It will now show up as a dependency but you will get a few more **cannot-find-android-14** errors in the event log.
6. Look in the **build.gradle** file for your app module. There are multiple build.gradle files in an Android project.

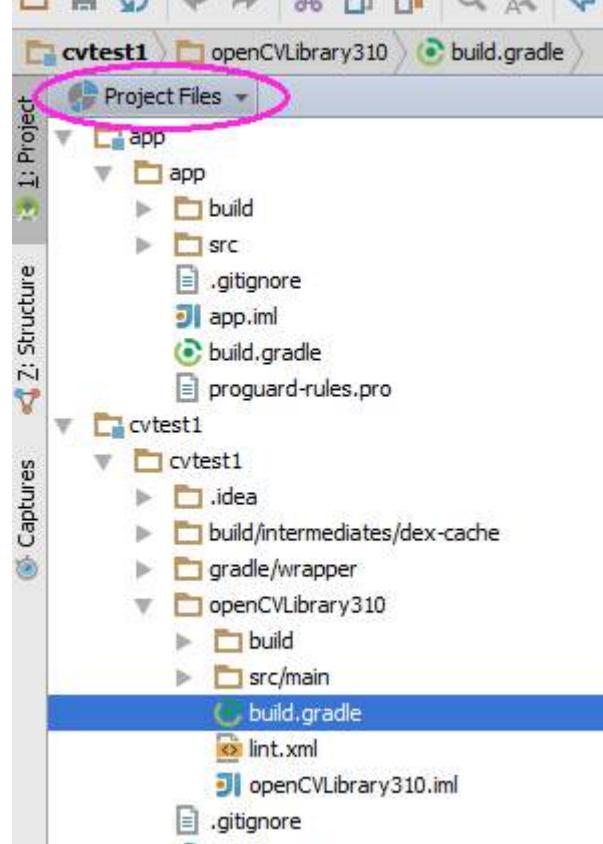
你需要的文件在**cvtest1/app**目录下，从项目视图看起来像是**build.gradle**（模块：app）。请注意这四个字段的值：

- compileSdkVersion (我的显示为23)
- buildToolsVersion (我的显示为23.0.2)
- minSdkVersion (我的显示为19)
- targetSdkVersion (我的显示为23)

7. 你的项目现在有一个**cvtest1/OpenCVLibrary310**目录，但在项目视图中不可见：



使用其他工具，比如任何文件管理器，进入该目录。你也可以将项目视图从Android切换到Project Files，就能找到该目录，如下图所示：



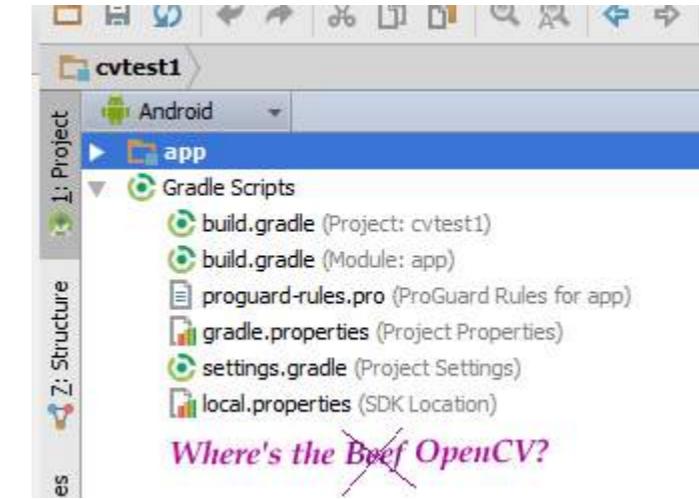
里面还有一个**build.gradle**文件（在上图中已高亮）。用第6步中的四个值更新此文件。

8. 重新同步你的项目，然后清理/重建它。（菜单：/Build/Clean_Project）它应该能清理并构建

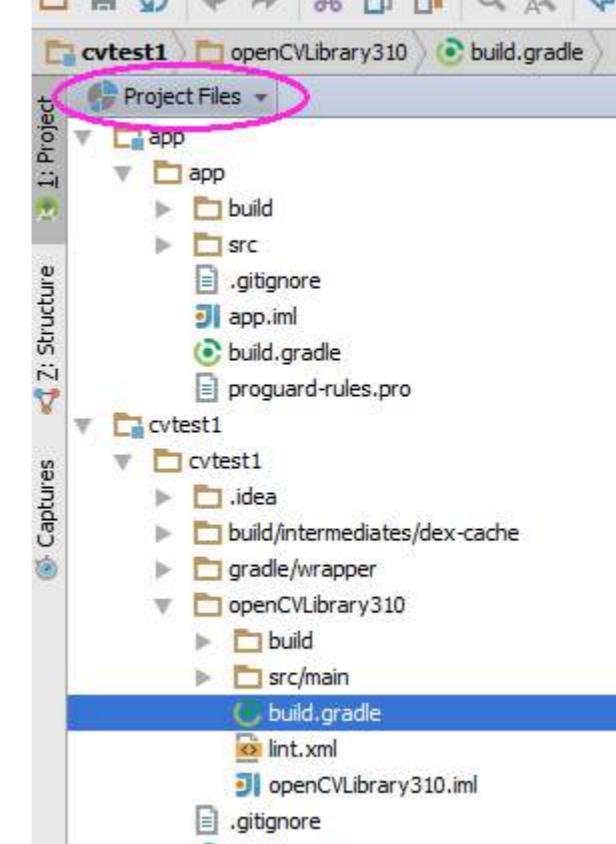
The one you want is in the **cvtest1/app** directory and from the project view it looks like **build.gradle** (Module: app). Note the values of these four fields:

- compileSdkVersion (mine says 23)
- buildToolsVersion (mine says 23.0.2)
- minSdkVersion (mine says 19)
- targetSdkVersion (mine says 23)

7. Your project now has a **cvtest1/OpenCVLibrary310** directory but it is not visible from the project view:



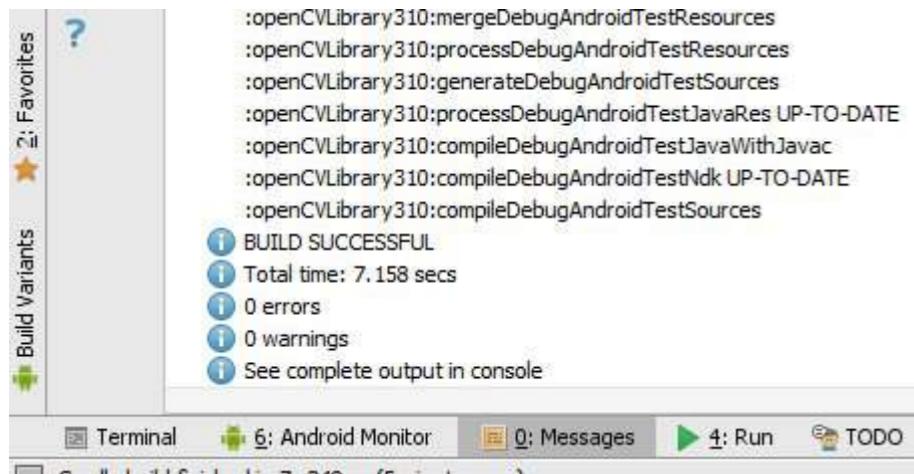
Use some other tool, such as any file manager, and go to this directory. You can also switch the project view from **Android** to **Project Files** and you can find this directory as shown in this screenshot:



Inside there is another **build.gradle** file (it's highlighted in the above screenshot). Update this file with the four values from step 6.

8. Resynch your project and then clean/rebuild it. (**Menu:/Build/Clean_Project**) It should clean and build

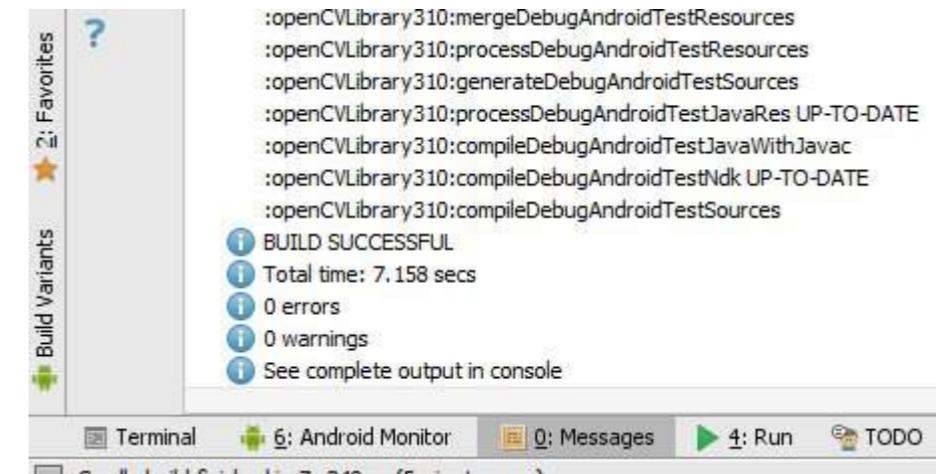
没有错误，你应该会在0:Messages屏幕中看到许多关于:openCVLibrary310的引用。



此时，模块应该会像app一样出现在项目层级中，名称为openCVLibrary310。（注意在那个小下拉菜单中，我从Project View切换回了Android View）。你还应该在“Gradle Scripts”下看到一个额外的build.gradle文件，但我发现Android Studio界面有点卡顿，有时不会立即显示。所以尝试重新同步、清理，甚至重启Android Studio。

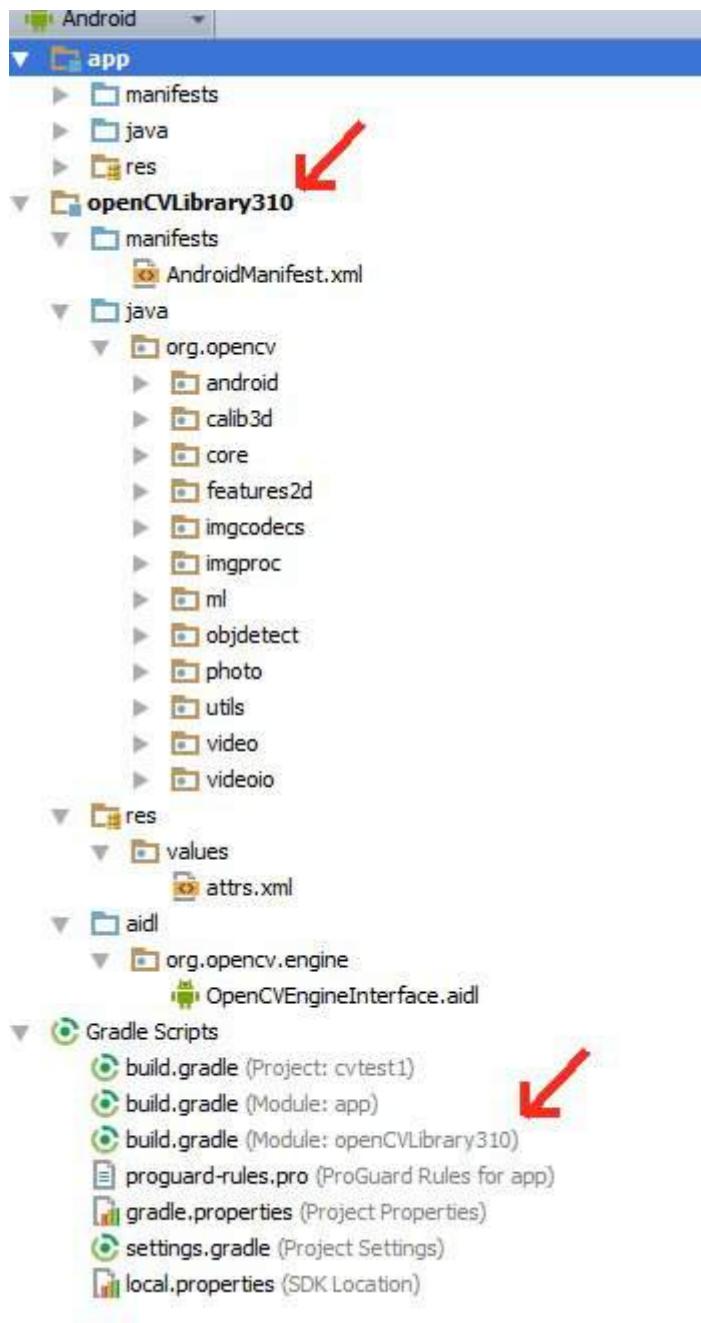
你应该能看到openCVLibrary310模块，里面包含所有OpenCV函数，位于java目录下，如下截图所示：

without errors and you should see many references to :openCVLibrary310 in the 0:Messages screen.

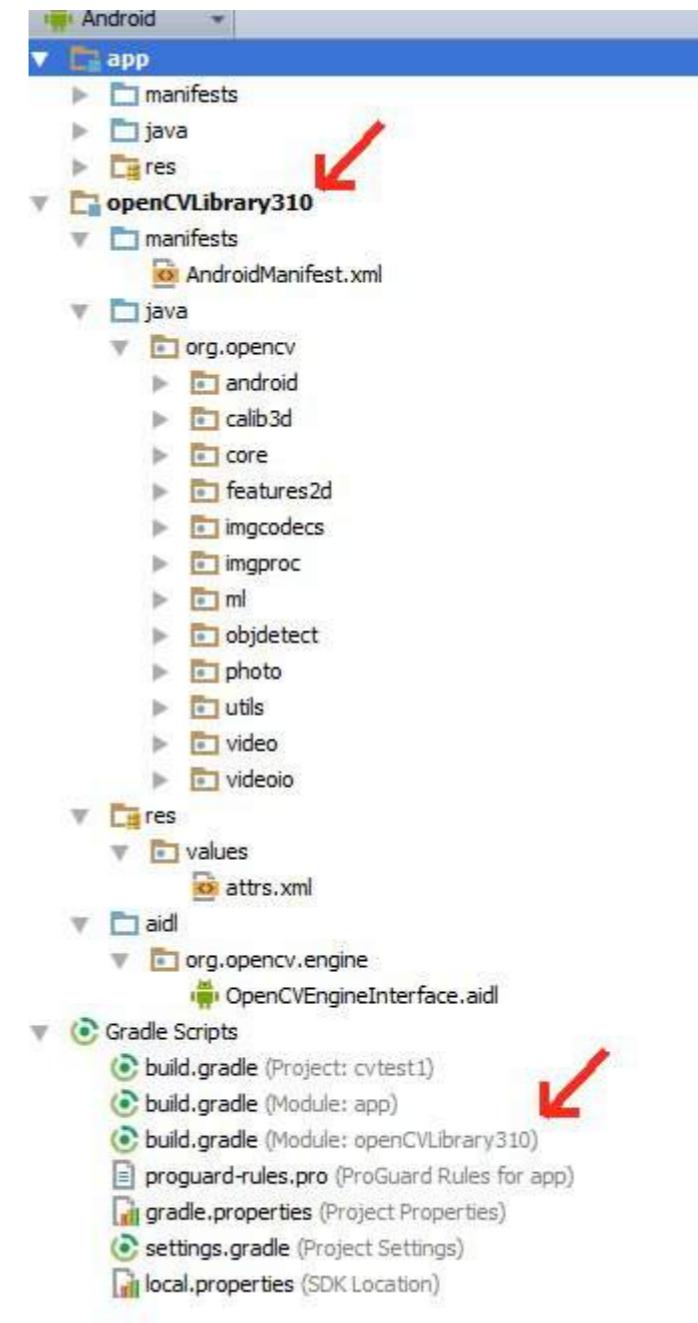


At this point the module should appear in the project hierarchy as **openCVLibrary310**, just like **app**. (Note that in that little drop-down menu I switched back from **Project View** to **Android View**). You should also see an additional **build.gradle** file under "Gradle Scripts" but I find the Android Studio interface a little bit glitchy and sometimes it does not do this right away. So try resynching, cleaning, even restarting Android Studio.

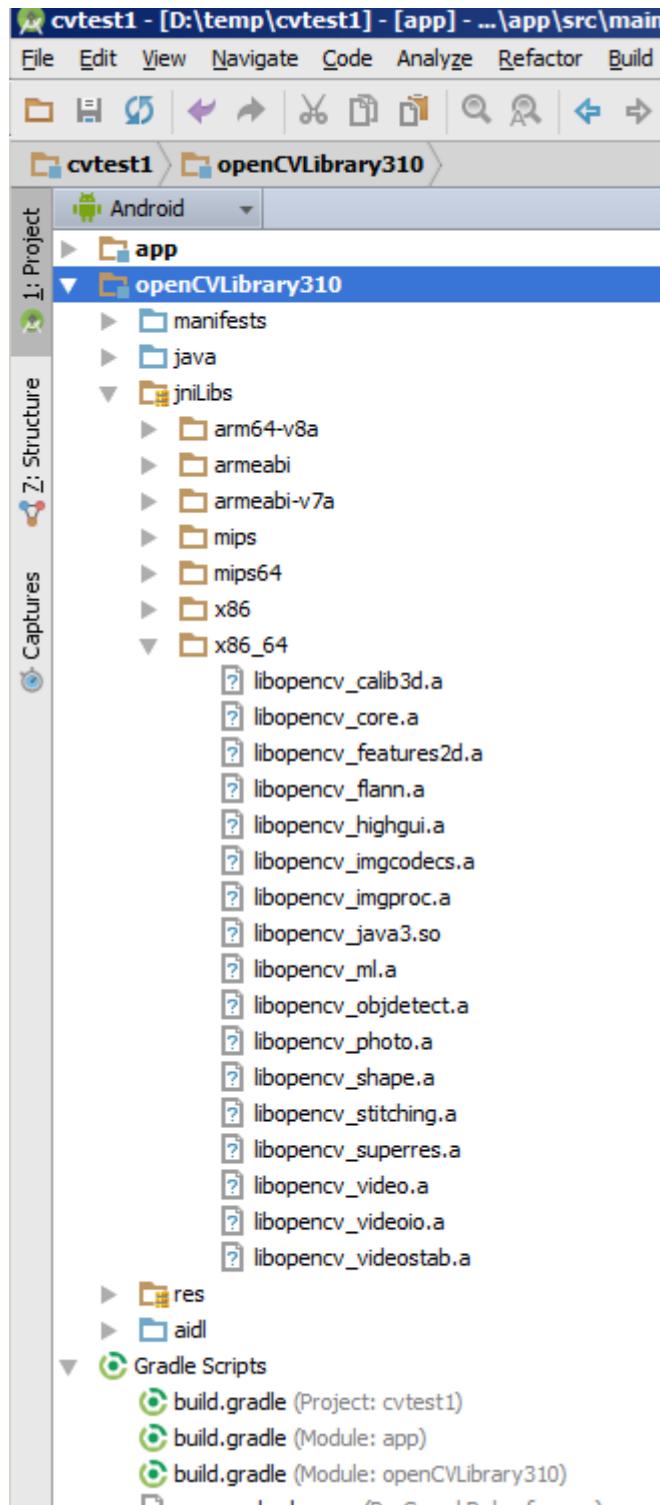
You should see the openCVLibrary310 module with all the OpenCV functions under java like in this screenshot:



9. 将{unzip-dir}/sdk/native/libs目录（及其所有内容）复制到你的Android项目中，路径为
cvtest1/OpenCVLibrary310/src/main/，然后将复制的libs重命名为jniLibs。现在你应该有一个cvtest1/OpenC
VLibrary310/src/main/jniLibs目录。重新同步你的项目，这个目录应该会出现在openCVLibrary310的项目视
图中。



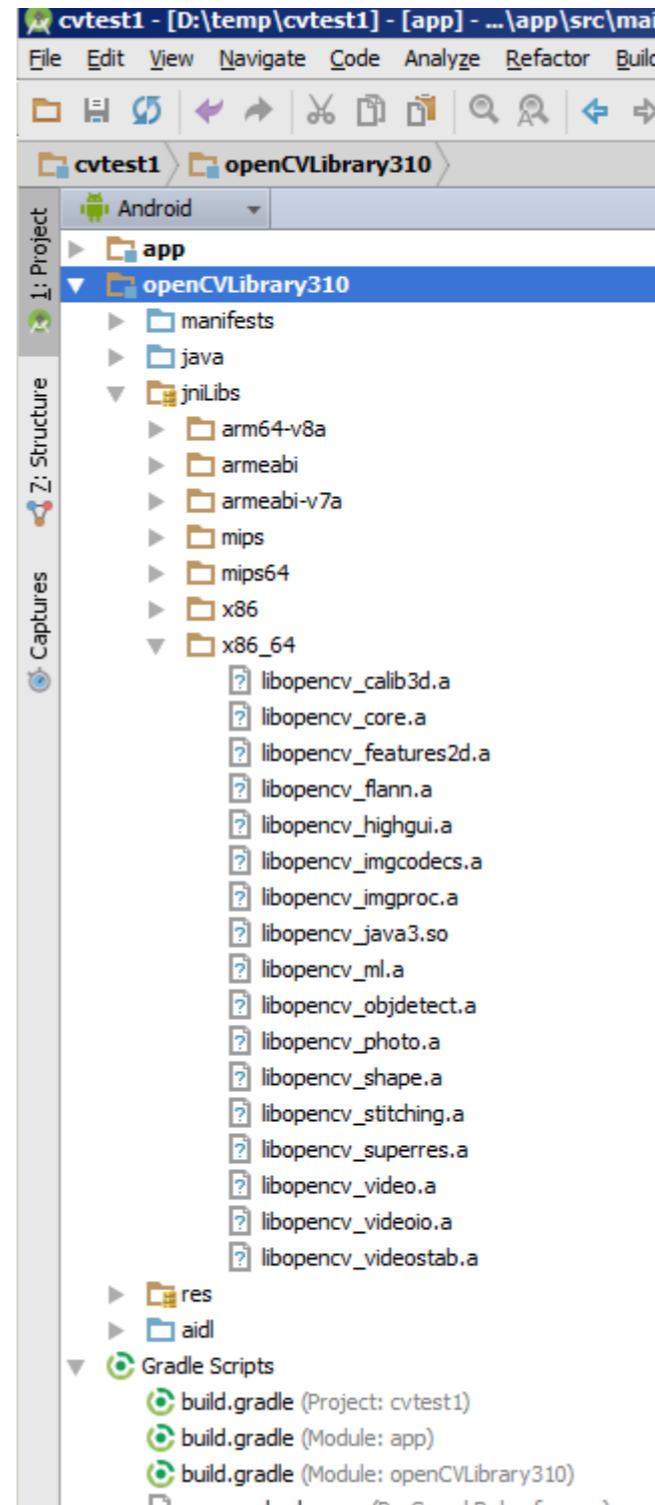
9. Copy the **{unzip-dir}/sdk/native/libs** directory (and everything under it) to your Android project, to
cvtest1/OpenCVLibrary310/src/main/, and then rename your copy from **libs** to **jniLibs**. You should now
have a **cvtest1/OpenCVLibrary310/src/main/jniLibs** directory. Resynch your project and this directory
should now appear in the project view under **openCVLibrary310**.



10. 进入MainActivity.java的onCreate方法，追加以下代码：

```
if (!OpenCVLoader.initDebug()) {
    Log.e(this.getClass().getSimpleName(), " OpenCVLoader.initDebug(), 未能正常工作。");
} else {
    Log.d(this.getClass().getSimpleName(), " OpenCVLoader.initDebug(), 正常工作。");
}
```

然后运行你的应用程序。你应该会在Android监视器中看到类似这样的行：



10. Go to the onCreate method of *MainActivity.java* and append this code:

```
if (!OpenCVLoader.initDebug()) {
    Log.e(this.getClass().getSimpleName(), " OpenCVLoader.initDebug(), not working.");
} else {
    Log.d(this.getClass().getSimpleName(), " OpenCVLoader.initDebug(), working.");
}
```

Then run your application. You should see lines like this in the Android Monitor:

```

4.4 - API 19 - 768x1280 Android 4.4.4 (API 19) No Debuggable Applications
GPU Network Log level: Verbose Show
4059-14059/? D/dalvikvm: VFY: replacing opcode 0x6e at 0x0002
4059-14059/? D/OpenCV/StaticHelper: Trying to get library list
4059-14059/? E/OpenCV/StaticHelper: OpenCV error: Cannot load info library for OpenCV
4059-14059/? D/OpenCV/StaticHelper: Library list: ""
4059-14059/? D/OpenCV/StaticHelper: First attempt to load libs
4059-14059/? D/OpenCV/StaticHelper: Trying to init OpenCV libs
4059-14059/? D/OpenCV/StaticHelper: Trying to load library opencv_java3
4059-14059/? D/dalvikvm: Trying to load lib /data/app-lib/com.imgur.cvtest1-2/libopencv_java3.so 0xa5051a78
11-655/? D/MobileDataStateTracker: default: setPolicyDataEnable(enabled=true)
4059-14059/? D/dalvikvm: Added shared lib /data/app-lib/com.imgur.cvtest1-2/libopencv_java3.so 0xa5051a78
4059-14059/? D/OpenCV/StaticHelper: Library opencv_java3 loaded
4059-14059/? D/OpenCV/StaticHelper: First attempt to load libs is OK
4059-14059/? I/OpenCV/StaticHelper: General configuration for OpenCV 3.1.0 =====
4059-14059/? I/OpenCV/StaticHelper: Version control: 3.1.0
4059-14059/? I/OpenCV/StaticHelper: Platform:
4059-14059/? I/OpenCV/StaticHelper: Host: Darwin 15.0.0 x86_64
4059-14059/? I/OpenCV/StaticHelper: Target: Android 1 i686
4059-14059/? I/OpenCV/StaticHelper: CMake: 3.3.2
4059-14059/? I/OpenCV/StaticHelper: CMake generator: Ninja
4059-14059/? I/OpenCV/StaticHelper: CMake build tool: /usr/local/bin/ninja
4059-14059/? I/OpenCV/StaticHelper: Configuration: Release
4059-14059/? I/OpenCV/StaticHelper: C/C++:
4059-14059/? I/OpenCV/StaticHelper: Built as dynamic libs?: NO
4059-14059/? I/OpenCV/StaticHelper: C++ Compiler: /usr/local/bin/ccache /opt/android/android-ndk-r10
4059-14059/? I/OpenCV/StaticHelper: C/C++ Flags /DCompiler...

```

(我不知道为什么那行错误信息会出现)

11. 现在尝试实际使用一些OpenCV代码。下面的示例中，我将一个.jpg文件复制到了Android模拟器上cvtest1应用的缓存目录中。下面的代码加载这张图片，运行Canny边缘检测算法，然后将结果写回到同一目录下的.png文件中。

将这段代码放在上一步代码的正下方，并根据你自己的文件/目录进行修改。

```

String inputFileName="simm_01";
String inputExtension = "jpg";
String inputDir = getCacheDir().getAbsolutePath(); // 使用缓存目录进行输入输出
String outputDir = getCacheDir().getAbsolutePath();
String outputExtension = "png";
String inputFilePath = inputDir + File.separator + inputFileName + "." + inputExtension;

```

```

Log.d (this.getClass().getSimpleName(), "loading " + inputFilePath + "...");
Mat image = Imgcodecs.imread(inputFilePath);
Log.d (this.getClass().getSimpleName(), "width of " + inputFileName + ":" + image.width());
// 如果宽度为0，则表示未能读取图像。

```

```

// 对于Canny边缘检测算法，可以调整这些参数以观察不同的结果
int threshold1 = 70;
int threshold2 = 100;

```

```

Mat im_canny = new Mat(); // 在传递给Canny方法之前必须初始化输出图像
Imgproc.Canny(image, im_canny, threshold1, threshold2);
String cannyFilename = outputDir + File.separator + inputFileName + "_canny-" + threshold1 + "-"
+ threshold2 + "." + outputExtension;
Log.d (this.getClass().getSimpleName(), "正在写入 " + cannyFilename);
Imgcodecs.imwrite(cannyFilename, im_canny);

```

```

4.4 - API 19 - 768x1280 Android 4.4.4 (API 19) No Debuggable Applications
GPU Network Log level: Verbose Show
4059-14059/? D/dalvikvm: VFY: replacing opcode 0x6e at 0x0002
4059-14059/? D/OpenCV/StaticHelper: Trying to get library list
4059-14059/? E/OpenCV/StaticHelper: OpenCV error: Cannot load info library for OpenCV
4059-14059/? D/OpenCV/StaticHelper: Library list: ""
4059-14059/? D/OpenCV/StaticHelper: First attempt to load libs
4059-14059/? D/OpenCV/StaticHelper: Trying to init OpenCV libs
4059-14059/? D/OpenCV/StaticHelper: Trying to load library opencv_java3
4059-14059/? D/dalvikvm: Trying to load lib /data/app-lib/com.imgur.cvtest1-2/libopencv_java3.so 0xa5051a78
11-655/? D/MobileDataStateTracker: default: setPolicyDataEnable(enabled=true)
4059-14059/? D/dalvikvm: Added shared lib /data/app-lib/com.imgur.cvtest1-2/libopencv_java3.so 0xa5051a78
4059-14059/? D/OpenCV/StaticHelper: Library opencv_java3 loaded
4059-14059/? D/OpenCV/StaticHelper: First attempt to load libs is OK
4059-14059/? I/OpenCV/StaticHelper: General configuration for OpenCV 3.1.0 =====
4059-14059/? I/OpenCV/StaticHelper: Version control: 3.1.0
4059-14059/? I/OpenCV/StaticHelper: Platform:
4059-14059/? I/OpenCV/StaticHelper: Host: Darwin 15.0.0 x86_64
4059-14059/? I/OpenCV/StaticHelper: Target: Android 1 i686
4059-14059/? I/OpenCV/StaticHelper: CMake: 3.3.2
4059-14059/? I/OpenCV/StaticHelper: CMake generator: Ninja
4059-14059/? I/OpenCV/StaticHelper: CMake build tool: /usr/local/bin/ninja
4059-14059/? I/OpenCV/StaticHelper: Configuration: Release
4059-14059/? I/OpenCV/StaticHelper: C/C++:
4059-14059/? I/OpenCV/StaticHelper: Built as dynamic libs?: NO
4059-14059/? I/OpenCV/StaticHelper: C++ Compiler: /usr/local/bin/ccache /opt/android/android-ndk-r10
4059-14059/? I/OpenCV/StaticHelper: C/C++ Flags /DCompiler...

```

(I don't know why that line with the error message is there)

11. Now try to actually use some openCV code. In the example below I copied a .jpg file to the cache directory of the cvtest1 application on the android emulator. The code below loads this image, runs the canny edge detection algorithm and then writes the results back to a .png file in the same directory.

Put **this** code just below the code from the previous step and alter it to match your own files/directories.

```

String inputFileName="simm_01";
String inputExtension = "jpg";
String inputDir = getCacheDir().getAbsolutePath(); // use the cache directory for i/o
String outputDir = getCacheDir().getAbsolutePath();
String outputExtension = "png";
String inputFilePath = inputDir + File.separator + inputFileName + "." + inputExtension;

```

```

Log.d (this.getClass().getSimpleName(), "loading " + inputFilePath + "...");
Mat image = Imgcodecs.imread(inputFilePath);
Log.d (this.getClass().getSimpleName(), "width of " + inputFileName + ":" + image.width());
// if width is 0 then it did not read your image.

```

```

// for the canny edge detection algorithm, play with these to see different results
int threshold1 = 70;
int threshold2 = 100;

```

```

Mat im_canny = new Mat(); // you have to initialize output image before giving it to the Canny method
Imgproc.Canny(image, im_canny, threshold1, threshold2);
String cannyFilename = outputDir + File.separator + inputFileName + "_canny-" + threshold1 + "-"
+ threshold2 + "." + outputExtension;
Log.d (this.getClass().getSimpleName(), "Writing " + cannyFilename);
Imgcodecs.imwrite(cannyFilename, im_canny);

```

12. 运行你的应用程序。你的模拟器应该会创建一个黑白的“边缘”图像。你可以使用 Android 设备监视器来获取输出，或者编写一个活动来显示它。

12. Run your application. Your emulator should create a black and white "edge" image. You can use the Android Device Monitor to retrieve the output or write an activity to show it.

第174章：MVVM（架构）

第174.1节：使用数据绑定库的MVVM示例

MVVM的核心目的是将包含逻辑的层与视图层分离。

在Android上，我们可以使用数据绑定库来帮助实现这一点，并使大部分逻辑可进行单元测试，而无需担心Android依赖。

在这个示例中，我将展示一个非常简单的应用的核心组件，该应用实现以下功能：

- 启动时模拟网络调用并显示加载旋转图标
- 显示一个包含点击计数器TextView、消息TextView和一个用于增加计数器的按钮的视图
- 点击按钮时更新计数器，并在计数器达到某个数值时更新计数器颜色和消息文本

让我们从视图层开始：

activity_main.xml:

如果你不熟悉数据绑定的工作原理，建议花10分钟时间熟悉一下。正如你所见，通常通过setter更新的所有字段都绑定到了viewModel变量上的函数。

如果你对android:visibility或app:textColor属性有疑问，请查看“备注”部分。

```
<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <import type="android.view.View" />

        <variable
            name="viewModel"
            type="de.walled.mvvmtest.viewmodel.ClickerViewModel"/>
    </data>

    <RelativeLayout
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="@dimen/activity_horizontal_margin"

        tools:context="de.walled.mvvmtest.view.MainActivity">

        <LinearLayout
            android:id="@+id/click_counter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="60dp"
            android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

            android:padding="8dp"

            android:orientation="horizontal">
```

Chapter 174: MVVM (Architecture)

Section 174.1: MVVM Example using DataBinding Library

The whole point of MVVM is to separate layers containing logic from the view layer.

On Android we can use the [DataBinding Library](#) to help us with this and make most of our logic Unit-testable without worrying about Android dependencies.

In this example I'll show the central components for a stupid simple App that does the following:

- At start up fake a network call and show a loading spinner
- Show a view with a click counter TextView, a message TextView, and a button to increment the counter
- On button click update counter and update counter color and message text if counter reaches some number

Let's start with the view layer:

activity_main.xml:

If you're unfamiliar with how DataBinding works you should probably [take 10 minutes](#) to make yourself familiar with it. As you can see, all fields you would usually update with setters are bound to functions on the viewModel variable.

If you've got a question about the android:visibility or app:textColor properties check the 'Remarks' section.

```
<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <import type="android.view.View" />

        <variable
            name="viewModel"
            type="de.walled.mvvmtest.viewmodel.ClickerViewModel"/>
    </data>

    <RelativeLayout
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="@dimen/activity_horizontal_margin"

        tools:context="de.walled.mvvmtest.view.MainActivity">

        <LinearLayout
            android:id="@+id/click_counter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="60dp"
            android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

            android:padding="8dp"

            android:orientation="horizontal">
```

```

<?>TextView
    android:id="@+id/number_of_clicks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/ClickCounter"

    android:text="@{viewModel.numberOfClicks}"
    android:textAlignment="center"
    app:textColor="@{viewModel.counterColor}"

    tools:text="8"
    tools:textColor="@color/red"
/>

<?>TextView
    android:id="@+id/static_label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="4dp"
    android:layout_marginStart="4dp"
    style="@style/ClickCounter"

    android:text="@string/label.clicks"
    app:textColor="@{viewModel.counterColor}"
    android:textAlignment="center"

    tools:textColor="@color/red"
/>
</LinearLayout>

```

```

<?>TextView
    android:id="@+id/message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/click_counter"
    android:layout_centerHorizontal="true"
    android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

    android:text="@{viewModel.labelText}"
    android:textAlignment="center"
    android:textSize="18sp"

    tools:text="You're bad and you should feel bad!"
/>

<Button
    android:id="@+id/clicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/message"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="8dp"
    android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

    android:padding="8dp"

    android:text="@string/label.button"

    android:onClick="@{() -> viewModel.onClickIncrement()}"
/>

```

```

<?>TextView
    android:id="@+id/number_of_clicks"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/ClickCounter"

    android:text="@{viewModel.numberOfClicks}"
    android:textAlignment="center"
    app:textColor="@{viewModel.counterColor}"

    tools:text="8"
    tools:textColor="@color/red"
/>

<?>TextView
    android:id="@+id/static_label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="4dp"
    android:layout_marginStart="4dp"
    style="@style/ClickCounter"

    android:text="@string/label.clicks"
    app:textColor="@{viewModel.counterColor}"
    android:textAlignment="center"

    tools:textColor="@color/red"
/>
</LinearLayout>

<?>TextView
    android:id="@+id/message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/click_counter"
    android:layout_centerHorizontal="true"
    android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

    android:text="@{viewModel.labelText}"
    android:textAlignment="center"
    android:textSize="18sp"

    tools:text="You're bad and you should feel bad!"
/>

<Button
    android:id="@+id/clicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/message"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="8dp"
    android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

    android:padding="8dp"

    android:text="@string/label.button"

    android:onClick="@{() -> viewModel.onClickIncrement()}"
/>

```

```

<android.support.v4.widget.ContentLoadingProgressBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="90dp"
    android:layout_centerHorizontal="true"
    style="@android:style/Widget.ProgressBar.Inverse"
    android:visibility="@{viewModel.loadingVisible ? View.VISIBLE : View.GONE}"

    android:indeterminate="true"
/>

</RelativeLayout>

</layout>

```

接下来是模型层。这里我有：

- 两个表示应用状态的字段
- 用于读取点击次数和激动状态的getter方法
- 一个用于增加点击次数的方法
- 一个用于恢复之前状态的方法（对屏幕方向变化很重要）

我还在这里定义了一个“兴奋状态”，它取决于点击次数。稍后将用于更新视图上的颜色和消息。

需要注意的是，模型中没有对状态如何显示给用户做出任何假设！

ClickerModel.java

```

import com.google.common.base.Optional;

import de.wallel.mvvmtest.viewmodel.ViewState;

public class ClickerModel implements IClickerModel {

    private int 点击次数;
    private Excitemen 兴奋状态;

    public void 增加点击次数() {
        点击次数 += 1;
        更新兴奋状态();
    }

    public int 获取点击次数() {
        return Optional.fromNullable(点击次数).or(0);
    }

    public Excitemen 获取兴奋状态() {
        return Optional.fromNullable(兴奋状态).or(Excitemen.BOO);
    }

    public void 恢复状态(ViewState 状态) {
        点击次数 = 状态.获取点击次数();
        更新兴奋状态();
    }

    private void updateStateOfExcitement() {
        if (numberOfClicks < 10) {
            stateOfExcitement = Excitemen.BOO;
        }
    }
}

```

```

<android.support.v4.widget.ContentLoadingProgressBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="90dp"
    android:layout_centerHorizontal="true"
    style="@android:style/Widget.ProgressBar.Inverse"
    android:visibility="@{viewModel.loadingVisible ? View.VISIBLE : View.GONE}"

    android:indeterminate="true"
/>

</RelativeLayout>

</layout>

```

Next the model layer. Here I have:

- two fields that represent the state of the app
- getters to read the number of clicks and the state of excitement
- a method to increment my click count
- a method to restore some previous state (important for orientation changes)

Also I define here a 'state of excitement' that is dependent on the number of clicks. This will later be used to update color and message on the View.

It is important to note that there are no assumptions made in the model about how the state might be displayed to the user!

ClickerModel.java

```

import com.google.common.base.Optional;

import de.wallel.mvvmtest.viewmodel.ViewState;

public class ClickerModel implements IClickerModel {

    private int numberofClicks;
    private Excitemen stateofExcitement;

    public void incrementClicks() {
        numberofClicks += 1;
        updateStateofExcitement();
    }

    public int getNumberofClicks() {
        return Optional.fromNullable(numberofClicks).or(0);
    }

    public Excitemen getStateofExcitement() {
        return Optional.fromNullable(stateofExcitement).or(Excitemen.BOO);
    }

    public void restoreState(ViewState state) {
        numberofClicks = state.getNumberofClicks();
        updateStateofExcitement();
    }

    private void updateStateofExcitement() {
        if (numberofClicks < 10) {
            stateofExcitement = Excitemen.BOO;
        }
    }
}

```

```

} else if (numberOfClicks <= 20) {
    stateOfExcitement = Excitement.MEH;
} else {
stateOfExcitement = Excitement.WOOHOO;
}
}
}
}

```

接下来是 ViewModel。

这将触发模型的变化，并将模型中的数据格式化以显示在视图上。请注意，正是在这里我们评估哪种 GUI 表示适合模型给出的状态 (`resolveCounterColor` 和 `resolveLabelText`)。因此，例如我们可以轻松实现一个 `UnderachieverClickerModel`，它对兴奋状态的阈值更低，而无需修改 `viewModel` 或视图中的任何代码。

还要注意，`ViewModel` 不持有任何视图对象的引用。所有属性都是通过 `@Bindable` 注解绑定的，并在调用 `notifyChange()`（表示所有属性需要更新）或 `notifyPropertyChanged(BR.propertyName)`（表示该属性需要更新）时进行更新。

`ClickerViewModel.java`

```

import android.databinding.BaseObservable;
import android.databinding.Bindable;
import android.support.annotation.ColorRes;
import android.support.annotation.StringRes;

import com.android.databinding.library.baseAdapters.BR;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.model.Excitement;
import de.walled.mvvmtest.model.IClickerModel;
import rx.Observable;

public class ClickerViewModel extends BaseObservable {

    private final IClickerApi api;
    boolean isLoading = false;
    private IClickerModel model;

    public ClickerViewModel(IClickerModel model, IClickerApi api) {
        this.model = model;
        this.api = api;
    }

    public void onClickIncrement() {
        model.incrementClicks();
        notifyChange();
    }

    public ViewState getViewState() {
        ViewState ViewState = new ViewState();
        ViewState.setNumberOfClicks(model.getNumberOfClicks());
        return ViewState;
    }

    public Observable<ViewState> loadData() {
        isLoading = true;
    }
}

```

```

} else if (numberOfClicks <= 20) {
    stateOfExcitement = Excitement.MEH;
} else {
    stateOfExcitement = Excitement.WOOHOO;
}
}
}
}

```

Next the `ViewModel`.

This will trigger changes on the model and format data from the model to show them on the view. Note that it is here where we evaluate which GUI representation is appropriate for the state given by the model (`resolveCounterColor` and `resolveLabelText`). So we could for example easily implement an `UnderachieverClickerModel` that has lower thresholds for the state of excitement without touching any code in the `viewModel` or view.

Also note that the `ViewModel` does not hold any references to view objects. All properties are bound via the `@Bindable` annotations and updated when either `notifyChange()` (signals all properties need to be updated) or `notifyPropertyChanged(BR.propertyName)` (signals this properties need to be updated).

`ClickerViewModel.java`

```

import android.databinding.BaseObservable;
import android.databinding.Bindable;
import android.support.annotation.ColorRes;
import android.support.annotation.StringRes;

import com.android.databinding.library.baseAdapters.BR;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.model.Excitement;
import de.walled.mvvmtest.model.IClickerModel;
import rx.Observable;

public class ClickerViewModel extends BaseObservable {

    private final IClickerApi api;
    boolean isLoading = false;
    private IClickerModel model;

    public ClickerViewModel(IClickerModel model, IClickerApi api) {
        this.model = model;
        this.api = api;
    }

    public void onClickIncrement() {
        model.incrementClicks();
        notifyChange();
    }

    public ViewState getViewState() {
        ViewState ViewState = new ViewState();
        ViewState.setNumberOfClicks(model.getNumberOfClicks());
        return ViewState;
    }

    public Observable<ViewState> loadData() {
        isLoading = true;
    }
}

```

```

    return api.fetchInitialState()
    .doOnNext(this::initModel)
        .doOnTerminate(() -> {
            isLoading = false;
            notifyPropertyChanged(BR.loadingVisible);
            notifyPropertyChanged(BR.contentVisible);
        });
}

public void initFromSavedState(ViewState savedState) {
    initModel(savedState);
}

@Bindable
public String getNumberOfClicks() {
    final int clicks = model.getNumberOfClicks();
    return String.valueOf(clicks);
}

@Bindable
@StringRes
public int getLabelText() {
    final Excitemen stateOfExcitemen = model.getStateOfExcitemen();
    return resolveLabelText(stateOfExcitemen);
}

@Bindable
@ColorRes
public int getCounterColor() {
    final Excitemen stateOfExcitemen = model.getStateOfExcitemen();
    return resolveCounterColor(stateOfExcitemen);
}

@Bindable
public boolean isLoadingVisible() {
    return isLoading;
}

@Bindable
public boolean isContentVisible() {
    return !isLoading;
}

private void initModel(final ViewState viewState) {
    model.restoreState(viewState);
    notifyChange();
}

@ColorRes
private int resolveCounterColor(Excitemen stateOfExcitemen) {
    switch (stateOfExcitemen) {
        case MEH:
            return R.color.yellow;
        case WOOHOO:
            return R.color.green;
        default:
            return R.color.red;
    }
}

@StringRes
private int resolveLabelText(Excitemen stateOfExcitemen) {

```

```

    return api.fetchInitialState()
    .doOnNext(this::initModel)
        .doOnTerminate(() -> {
            isLoading = false;
            notifyPropertyChanged(BR.loadingVisible);
            notifyPropertyChanged(BR.contentVisible);
        });
}

public void initFromSavedState(ViewState savedState) {
    initModel(savedState);
}

@Bindable
public String getNumberOfClicks() {
    final int clicks = model.getNumberOfClicks();
    return String.valueOf(clicks);
}

@Bindable
@StringRes
public int getLabelText() {
    final Excitemen stateOfExcitemen = model.getStateOfExcitemen();
    return resolveLabelText(stateOfExcitemen);
}

@Bindable
@ColorRes
public int getCounterColor() {
    final Excitemen stateOfExcitemen = model.getStateOfExcitemen();
    return resolveCounterColor(stateOfExcitemen);
}

@Bindable
public boolean isLoadingVisible() {
    return isLoading;
}

@Bindable
public boolean isContentVisible() {
    return !isLoading;
}

private void initModel(final ViewState viewState) {
    model.restoreState(viewState);
    notifyChange();
}

@ColorRes
private int resolveCounterColor(Excitemen stateOfExcitemen) {
    switch (stateOfExcitemen) {
        case MEH:
            return R.color.yellow;
        case WOOHOO:
            return R.color.green;
        default:
            return R.color.red;
    }
}

@StringRes
private int resolveLabelText(Excitemen stateOfExcitemen) {

```

```

switch (stateOfExcitement) {
    case MEH:
        return R.string.label_indifferent;
    case WOOHOO:
        return R.string.label_excited;
    default:
        return R.string.label_negative;
}
}
}

```

将所有内容整合到Activity中！

这里我们看到视图正在使用所有可能需要的依赖项初始化 viewModel，这些依赖项必须从安卓上下文中实例化。

在 viewModel 初始化后，它通过 DataBindingUtil 绑定到 xml 布局（请查看“语法”部分以了解生成类的命名）。

注意订阅是在此层进行订阅的，因为我们必须在活动暂停或销毁时处理取消订阅，以避免内存泄漏和空指针异常。同时，视图状态在屏幕方向变化时的持久化和重新加载也在这里触发。

MainActivity.java

```

import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.ClickerApi;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.databinding.ActivityMainBinding;
import de.walled.mvvmtest.model.ClickerModel;
import de.walled.mvvmtest.viewmodel.ClickerViewModel;
import de.walled.mvvmtest.viewmodel.ViewState;
import rx.Subscription;
import rx.subscriptions.Subscriptions;

public class MainActivity extends AppCompatActivity {

    private static final String KEY_VIEW_STATE = "state.view";

    private ClickerViewModel viewModel;
    private Subscription fakeLoader = Subscriptions.unsubscribed();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 通常会被注入，但我觉得 Dagger 超出范围
        final IClickerApi api = new ClickerApi();
        setupViewModel(savedInstanceState, api);

        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
        binding.setViewModel(viewModel);
    }

    @Override

```

```

switch (stateOfExcitement) {
    case MEH:
        return R.string.label_indifferent;
    case WOOHOO:
        return R.string.label_excited;
    default:
        return R.string.label_negative;
}
}
}

```

Tying it all together in the Activity!

Here we see the view initializing the viewModel with all dependencies it might need, that have to be instantiated from an android context.

After the viewModel is initialized it is bound to the xml layout via the DataBindingUtil (Please check 'Syntax' section for naming of generated classes).

Note subscriptions are subscribed to on this layer because we have to handle unsubscribing them when the activity is paused or destroyed to avoid memory leaks and NPEs. Also persisting and reloading of the ViewState on OrientationChanges is triggered here

MainActivity.java

```

import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.ClickerApi;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.databinding.ActivityMainBinding;
import de.walled.mvvmtest.model.ClickerModel;
import de.walled.mvvmtest.viewmodel.ClickerViewModel;
import de.walled.mvvmtest.viewmodel.ViewState;
import rx.Subscription;
import rx.subscriptions.Subscriptions;

public class MainActivity extends AppCompatActivity {

    private static final String KEY_VIEW_STATE = "state.view";

    private ClickerViewModel viewModel;
    private Subscription fakeLoader = Subscriptions.unsubscribed();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // would usually be injected but I feel Dagger would be out of scope
        final IClickerApi api = new ClickerApi();
        setupViewModel(savedInstanceState, api);

        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
        binding.setViewModel(viewModel);
    }

    @Override

```

```

protected void onPause() {
    fakeLoader.unsubscribe();
    super.onPause();
}

@Override
protected void onDestroy() {
    fakeLoader.unsubscribe();
    super.onDestroy();
}

@Override
protected void onSaveInstanceState(Bundle outState) {
    outState.putSerializable(KEY_VIEW_STATE, viewModel.getViewState());
}

private void setupViewModel(Bundle savedInstanceState, IClickerApi api) {
    viewModel = new ClickerViewModel(new ClickerModel(), api);
    final ViewState savedState = getViewStateFromBundle(savedInstanceState);

    if (savedState == null) {
        fakeLoader = viewModel.loadData().subscribe();
    } else {
        viewModel.initFromSavedState(savedState);
    }
}

private ViewState getViewStateFromBundle(Bundle savedInstanceState) {
    if (savedInstanceState != null) {
        return (ViewState) savedInstanceState.getSerializable(KEY_VIEW_STATE);
    }
    return null;
}

```

要查看所有操作，请查看此示例项目。

```

protected void onPause() {
    fakeLoader.unsubscribe();
    super.onPause();
}

@Override
protected void onDestroy() {
    fakeLoader.unsubscribe();
    super.onDestroy();
}

@Override
protected void onSaveInstanceState(Bundle outState) {
    outState.putSerializable(KEY_VIEW_STATE, viewModel.getViewState());
}

private void setupViewModel(Bundle savedInstanceState, IClickerApi api) {
    viewModel = new ClickerViewModel(new ClickerModel(), api);
    final ViewState savedState = getViewStateFromBundle(savedInstanceState);

    if (savedState == null) {
        fakeLoader = viewModel.loadData().subscribe();
    } else {
        viewModel.initFromSavedState(savedState);
    }
}

private ViewState getViewStateFromBundle(Bundle savedInstanceState) {
    if (savedInstanceState != null) {
        return (ViewState) savedInstanceState.getSerializable(KEY_VIEW_STATE);
    }
    return null;
}

```

To see everything in action check out this [example project](#).

第175章：Android中的ORMLite

第175.1节：基于SQLite的Android OrmLite示例

ORMLite 是一个对象关系映射包，提供简单且轻量级的功能，用于将Java对象持久化到SQL数据库，同时避免了更标准ORM包的复杂性和开销。

对于Android来说，OrmLite是基于开箱即用支持的数据库SQLite实现的。它直接调用API来访问SQLite。

Gradle设置

要开始使用，您应该将该包包含到构建的gradle中。

```
// https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
```

POJO配置

然后您应该配置一个POJO以持久化到数据库。这里需要注意注解：

- 在每个类的顶部添加@DatabaseTable注解。您也可以使用@Entity。
- 在每个需要持久化的字段前添加@DatabaseField注解。您也可以使用@Column和其他注解。
- 为每个类添加一个至少具有包可见性的无参构造函数。

```
@DatabaseTable(tableName = "form_model")
public class FormModel implements Serializable {

    @DatabaseField(generatedId = true)
    private Long id;
    @DatabaseField(dataType = DataType.SERIALIZABLE)
    ArrayList<ReviewItem> reviewItems;

    @DatabaseField(index = true)
    private String username;

    @DatabaseField
    private String createdAt;

    public FormModel() {
    }

    public FormModel(ArrayList<ReviewItem> reviewItems, String username, String createdAt) {
        this.reviewItems = reviewItems;
        this.username = username;
        this.createdAt = createdAt;
    }
}
```

上面的示例中有一个表 (form_model) ，包含4个字段。

id字段是自动生成的索引。

用户名是数据库的索引。

有关注释的更多信息可以在官方文档中找到。

Chapter 175: ORMLite in android

Section 175.1: Android OrmLite over SQLite example

ORMLite is an Object Relational Mapping package that provides simple and lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages.

Speaking for Android, OrmLite is implemented over the out-of-the-box supported database, SQLite. It makes direct calls to the API to access SQLite.

Gradle setup

To get started you should include the package to the build gradle.

```
// https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
```

POJO configuration

Then you should configure a POJO to be persisted to the database. Here care must be taken to the annotations:

- Add the @DatabaseTable annotation to the top of each class. You can also use @Entity.
- Add the @DatabaseField annotation right before each field to be persisted. You can also use @Column and others.
- Add a no-argument constructor to each class with at least package visibility.

```
@DatabaseTable(tableName = "form_model")
public class FormModel implements Serializable {

    @DatabaseField(generatedId = true)
    private Long id;
    @DatabaseField(dataType = DataType.SERIALIZABLE)
    ArrayList<ReviewItem> reviewItems;

    @DatabaseField(index = true)
    private String username;

    @DatabaseField
    private String createdAt;

    public FormModel() {
    }

    public FormModel(ArrayList<ReviewItem> reviewItems, String username, String createdAt) {
        this.reviewItems = reviewItems;
        this.username = username;
        this.createdAt = createdAt;
    }
}
```

At the example above there is one table (form_model) with 4 fields.

id field is auto generated index.

username is an index to the database.

More information about the annotation can be found at the [official documentation](#).

数据库助手

接下来，您需要创建一个数据库助手类，该类应继承
OrmLiteSqliteOpenHelper类。

该类在您的应用安装时创建和升级数据库，还可以提供其他类使用的DAO类。

DAO代表数据访问对象，它提供所有的Scrum功能，并专门处理单个持久化类。

助手类必须实现以下两个方法：

- onCreate(SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

onCreate在您的应用首次安装时创建数据库

- onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource, int oldVersion, int newVersion);

onUpgrade 处理当您将应用升级到新版本时数据库表的升级

数据库助手类示例：

```
public class OrmLite extends OrmLiteSqliteOpenHelper {

    //数据库名称
    private static final String DATABASE_NAME = "gaia";
    //数据库版本。更改版本将调用 {@Link OrmLite.onUpgrade}
    private static final int DATABASE_VERSION = 2;

    /**
     * 用于与 Sqlite 数据库交互以执行增删改查操作的数据访问对象。
     */
    private Dao<FormModel, Long> todoDao;

    public OrmLite(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION,
            /**
             * R.raw.ormlite_config 是指向本项目
             * /res/raw/ 目录下 ormlite_config2.txt 文件的引用
             */
        R.raw.ormlite_config2);
    }

    @Override
    public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {
        try {
            /**
             * 创建数据库表
             */
            TableUtils.createTable(connectionSource, FormModel.class);

        } catch (SQLException e) {
    }
}
```

Database Helper

To continue with, you will need to create a database helper class which should extend the OrmLiteSqliteOpenHelper class.

This class creates and upgrades the database when your application is installed and can also provide the DAO classes used by your other classes.

DAO stands for Data Access Object and it provides all the scrum functionality and specializes in the handling a single persisted class.

The helper class must implement the following two methods:

- onCreate(SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

onCreate creates the database when your app is first installed

- onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource, int oldVersion, int newVersion);

onUpgrade handles the upgrading of the database tables when you upgrade your app to a new version

Database Helper class example:

```
public class OrmLite extends OrmLiteSqliteOpenHelper {

    //Database name
    private static final String DATABASE_NAME = "gaia";
    //Version of the database. Changing the version will call {@Link OrmLite.onUpgrade}
    private static final int DATABASE_VERSION = 2;

    /**
     * The data access object used to interact with the Sqlite database to do C.R.U.D operations.
     */
    private Dao<FormModel, Long> todoDao;

    public OrmLite(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION,
            /**
             * R.raw.ormlite_config is a reference to the ormlite_config2.txt file in the
             * /res/raw/ directory of this project
             */
        R.raw.ormlite_config2);
    }

    @Override
    public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {
        try {
            /**
             * creates the database table
             */
            TableUtils.createTable(connectionSource, FormModel.class);

        } catch (SQLException e) {
    }
}
```

```

e.printStackTrace();
    } catch (java.sql.SQLException e) {
e.printStackTrace();
}
/*
当你用比已打开数据库版本更新的版本构造SQLiteOpenHelper时会调用此方法。
*/
@Override
public void onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource,
                      int oldVersion, int newVersion) {
    try {
        /**
         * 当框架调用 onUpgrade 时重新创建数据库
         */
        TableUtils.dropTable(connectionSource, FormModel.class, false);
        onCreate(database, connectionSource);

    } catch (SQLException | java.sql.SQLException e) {
e.printStackTrace();
    }
}

/**
* 返回数据访问对象的实例
* @return
* @throws SQLException
*/
public Dao<FormModel, Long> getDao() throws SQLException {
    if(todoDao == null) {
        try {
todoDao = getDao(FormModel.class);
        } catch (java.sql.SQLException e) {
e.printStackTrace();
        }
    }
    return todoDao;
}

```

将对象持久化到 SQLite

最后，将对象持久化到数据库的类。

```

public class ReviewPresenter {
    Dao<FormModel, Long> simpleDao;

    public ReviewPresenter(Application application) {
        this.application = (GaiaApplication) application;
        simpleDao = this.application.getHelper().getDao();
    }

    public void storeFormToSqlite(FormModel form) {

        try {
simpleDao.create(form);
        } catch (SQLException e) {
e.printStackTrace();
        }
List<FormModel> list = null;

```

```

e.printStackTrace();
    } catch (java.sql.SQLException e) {
e.printStackTrace();
}
/*
It is called when you construct a SQLiteOpenHelper with version newer than the version of
the opened database.
*/
@Override
public void onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource,
                      int oldVersion, int newVersion) {
    try {
        /**
         * Recreates the database when onUpgrade is called by the framework
         */
        TableUtils.dropTable(connectionSource, FormModel.class, false);
        onCreate(database, connectionSource);

    } catch (SQLException | java.sql.SQLException e) {
e.printStackTrace();
    }
}

/**
* Returns an instance of the data access object
* @return
* @throws SQLException
*/
public Dao<FormModel, Long> getDao() throws SQLException {
    if(todoDao == null) {
        try {
todoDao = getDao(FormModel.class);
        } catch (java.sql.SQLException e) {
e.printStackTrace();
        }
    }
    return todoDao;
}

```

Persisting Object to SQLite

Finally, the class that persists the object to the database.

```

public class ReviewPresenter {
    Dao<FormModel, Long> simpleDao;

    public ReviewPresenter(Application application) {
        this.application = (GaiaApplication) application;
        simpleDao = this.application.getHelper().getDao();
    }

    public void storeFormToSqlite(FormModel form) {

        try {
simpleDao.create(form);
        } catch (SQLException e) {
e.printStackTrace();
        }
List<FormModel> list = null;

```

```

try {
// 查询数据库中所有的数据对象
list = simpleDao.queryForAll();
} catch (SQLException e) {
e.printStackTrace();
}
// 用于构建内容视图的字符串构建器
StringBuilder sb = new StringBuilder();
int simpleC = 1;
for (FormModel simple : list) {
sb.append('#').append(simpleC).append(": ").append(simple.getUsername()).append("\n");
simpleC++;
}
System.out.println(sb.toString());
}

//Query to database to get all forms by username
public List<FormModel> getAllFormsByUsername(String username) {
List<FormModel> results = null;
try {
results = simpleDao.queryBuilder().where().eq("username",
PreferencesManager.getInstance().getString(Constants.USERNAME)).query();
} catch (SQLException e) {
e.printStackTrace();
}
return results;
}

```

上述类构造函数中DOA的访问器定义为：

```

private OrmLite dbHelper = null;

/*
在应用程序中提供SQLite辅助对象
*/
public OrmLite 获取助手() {
    if (dbHelper == null) {
        dbHelper = OpenHelperManager.getHelper(this, OrmLite.class);
    }
    return dbHelper;
}

```

```

try {
// query for all of the data objects in the database
list = simpleDao.queryForAll();
} catch (SQLException e) {
e.printStackTrace();
}
// our string builder for building the content-view
StringBuilder sb = new StringBuilder();
int simpleC = 1;
for (FormModel simple : list) {
sb.append('#').append(simpleC).append(": ").append(simple.getUsername()).append('\n');
simpleC++;
}
System.out.println(sb.toString());
}

//Query to database to get all forms by username
public List<FormModel> getAllFormsByUsername(String username) {
List<FormModel> results = null;
try {
results = simpleDao.queryBuilder().where().eq("username",
PreferencesManager.getInstance().getString(Constants.USERNAME)).query();
} catch (SQLException e) {
e.printStackTrace();
}
return results;
}

```

The accessor of the DOA at the constructor of the above class is defined as:

```

private OrmLite dbHelper = null;

/*
Provides the SQLite Helper Object among the application
*/
public OrmLite getHelper() {
    if (dbHelper == null) {
        dbHelper = OpenHelperManager.getHelper(this, OrmLite.class);
    }
    return dbHelper;
}

```

第176章：Retrofit2 与 RxJava

第176.1节：Retrofit2 与 RxJava

首先，在 build.gradle 文件中添加相关依赖。

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.3.0'
    ...
}
```

然后创建你想要接收的模型：

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

创建一个包含与远程服务器交换数据方法的接口：

```
public interface ApiServerRequests {
    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

然后创建一个Retrofit实例：

```
public ApiRequests DeviceAPIHelper () {
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://example.com/")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
        .build();

    api = retrofit.create(ApiServerRequests.class);
    return api;
}
```

然后，在代码的任何地方，调用该方法：

```
apiRequests.getServers()
    .subscribeOn(Schedulers.io()) // 可观察对象在io线程发射
    .observerOn(AndroidSchedulers.mainThread()) // 处理后台请求所需的方法
    .subscribe(new Subscriber<List<Server>>() {
        @Override
        public void onCompleted() {
```

Chapter 176: Retrofit2 with RxJava

Section 176.1: Retrofit2 with RxJava

First, add relevant dependencies into the build.gradle file.

```
dependencies {
    ...
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.3.0'
    ...
}
```

Then create the model you would like to receive:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Create an interface containing methods used to exchange data with remote server:

```
public interface ApiServerRequests {
    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Then create a Retrofit instance:

```
public ApiRequests DeviceAPIHelper () {
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://example.com/")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
        .build();

    api = retrofit.create(ApiServerRequests.class);
    return api;
}
```

Then, anywhere from the code, call the method:

```
apiRequests.getServers()
    .subscribeOn(Schedulers.io()) // the observable is emitted on io thread
    .observerOn(AndroidSchedulers.mainThread()) // Methods needed to handle request in background
    .subscribe(new Subscriber<List<Server>>() {
        @Override
        public void onCompleted() {
```

```

    }

    @Override
    public void onError(Throwable e) {
    }

    @Override
    public void onNext(List<Server> servers) {
        // 成功获取服务器列表
    }
);

```

第176.2节：嵌套请求示例：多个请求， 合并结果

假设我们有一个API，允许我们通过单次请求（getAllPets）获取对象元数据，还有其他请求可以获取单个资源的完整数据（getSinglePet）。我们如何在一个链中查询所有这些数据？

```

public class PetsFetcher {

    static class PetRepository {
        List<Integer> ids;
    }

    static class Pet {
        int id;
        String name;
        int weight;
        int height;
    }

    interface PetApi {
        @GET("pets") Observable<PetRepository> getAllPets();

        @GET("pet/{id}") Observable<Pet> getSinglePet(@Path("id") int id);
    }

    PetApi petApi;

    Disposable petsDisposable;

    public void requestAllPets() {
        petApi.getAllPets()
            .doOnSubscribe(new Consumer<Disposable>() {
                @Override public void accept(Disposable disposable) throws Exception {
                    petsDisposable = disposable;
                }
            })
            .flatMap(new Function<PetRepository, ObservableSource<Integer>>() {
                @Override
                public ObservableSource<Integer> apply(PetRepository petRepository) throws
                Exception {
                    List<Integer> petIds = petRepository.ids;
                    return Observable.fromIterable(petIds);
                }
            });
    }
}

```

```

    }

    @Override
    public void onError(Throwable e) {
    }

    @Override
    public void onNext(List<Server> servers) {
        // A list of servers is fetched successfully
    }
);

```

Section 176.2: Nested requests example: multiple requests, combine results

Suppose we have an API which allows us to get object metadata in single request (getAllPets), and other request which have full data of single resource (getSinglePet). How we can query all of them in a single chain?

```

public class PetsFetcher {

    static class PetRepository {
        List<Integer> ids;
    }

    static class Pet {
        int id;
        String name;
        int weight;
        int height;
    }

    interface PetApi {
        @GET("pets") Observable<PetRepository> getAllPets();

        @GET("pet/{id}") Observable<Pet> getSinglePet(@Path("id") int id);
    }

    PetApi petApi;

    Disposable petsDisposable;

    public void requestAllPets() {
        petApi.getAllPets()
            .doOnSubscribe(new Consumer<Disposable>() {
                @Override public void accept(Disposable disposable) throws Exception {
                    petsDisposable = disposable;
                }
            })
            .flatMap(new Function<PetRepository, ObservableSource<Integer>>() {
                @Override
                public ObservableSource<Integer> apply(PetRepository petRepository) throws
                Exception {
                    List<Integer> petIds = petRepository.ids;
                    return Observable.fromIterable(petIds);
                }
            });
    }
}

```

```

        })
.flatMap(new Function<Integer, ObservableSource<Pet>>() {
    @Override public ObservableSource<Pet> apply(Integer id) throws Exception {
        return petApi.getSinglePet(id);
    }
})
.toList()
.toObservable()
.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThread())
.subscribe(new Consumer<List<Pet>>() {
    @Override public void accept(List<Pet> pets) throws Exception {
        //在这里使用你的宠物
    }
}, new Consumer<Throwable>() {
@Override public void accept(Throwable throwable) throws Exception {
    //向用户显示出错信息
}
});
}

void cancelRequests(){
if (petsDisposable!=null){
    petsDisposable.dispose();
    petsDisposable = null;
}
}
}

```

第176.3节：使用Retrofit和RxJava异步获取数据

来自RxJava的GitHub仓库，RxJava是Reactive Extensions的Java虚拟机实现：一个通过使用可观察序列来组合异步和基于事件的程序的库。它扩展了观察者模式以支持数据/事件序列，并添加了操作符，允许你声明式地组合序列，同时抽象掉诸如底层线程、同步、线程安全和并发数据结构等问题。

Retrofit是一个适用于Android和Java的类型安全HTTP客户端，使用它，开发者可以让所有网络操作变得更加简单。举例来说，我们将下载一些JSON并在RecyclerView中以列表形式展示。

入门：

在你的应用级build.gradle文件中添加RxJava、RxAndroid和Retrofit依赖： compile
 "io.reactivex:rxjava:1.1.6"
 compile "io.reactivex:rxbinding:1.2.1"
 compile "com.squareup.retrofit2:adapter-rxjava:2.0.2"
 compile "com.google.code.gson:gson:2.6.2"
 compile "com.squareup.retrofit2:retrofit:2.0.2"
 compile "com.squareup.retrofit2:converter-gson:2.0.2"

定义 ApiClient 和 ApiInterface 用于与服务器交换数据

```

public class ApiClient {

private static Retrofit retrofitInstance = null;
private static final String BASE_URL = "https://api.github.com/";

```

```

        })
.flatMap(new Function<Integer, ObservableSource<Pet>>() {
    @Override public ObservableSource<Pet> apply(Integer id) throws Exception {
        return petApi.getSinglePet(id);
    }
})
.toList()
.toObservable()
.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThread())
.subscribe(new Consumer<List<Pet>>() {
    @Override public void accept(List<Pet> pets) throws Exception {
        //use your pets here
    }
}, new Consumer<Throwable>() {
@Override public void accept(Throwable throwable) throws Exception {
    //show user something goes wrong
}
});
}

void cancelRequests(){
if (petsDisposable!=null){
    petsDisposable.dispose();
    petsDisposable = null;
}
}
}

```

Section 176.3: Retrofit with RxJava to fetch data asynchronously

From the [GitHub repo](#) of RxJava, RxJava is a Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences. It extends the observer pattern to support sequences of data/events and adds operators that allow you to compose sequences together declaratively while abstracting away concerns about things like low-level threading, synchronisation, thread-safety and concurrent data structures.

[Retrofit](#) is a type-safe HTTP client for Android and Java, using this, developers can make all network stuff much more easier. As an example, we are going to download some JSON and show it in RecyclerView as a list.

Getting started:

Add RxJava, RxAndroid and Retrofit dependencies in your app level build.gradle file: compile
 "io.reactivex:rxjava:1.1.6"
 compile "io.reactivex:rxbinding:1.2.1"
 compile "com.squareup.retrofit2:adapter-rxjava:2.0.2"
 compile "com.google.code.gson:gson:2.6.2"
 compile "com.squareup.retrofit2:retrofit:2.0.2"
 compile "com.squareup.retrofit2:converter-gson:2.0.2"

Define ApiClient and ApiInterface to exchange data from server

```

public class ApiClient {

private static Retrofit retrofitInstance = null;
private static final String BASE_URL = "https://api.github.com/";

```

```

public static Retrofit getInstance() {
    if (retrofitInstance == null) {
        retrofitInstance = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .addConverterFactory(GsonConverterFactory.create())
        .build();
    }
    return retrofitInstance;
}

public static <T> T createRetrofitService(final Class<T> clazz, final String endPoint) {
    final Retrofit restAdapter = new Retrofit.Builder()
        .baseUrl(endPoint)
        .build();

    return restAdapter.create(clazz);
}

public static String getBaseUrl() {
    return BASE_URL;
}

```

public interface ApiInterface {

```

@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                         @Path("repo") String repositoryName,
                                         @Query("page") int pageNumber);}

```

注意，getRepos() 返回的是一个 Observable，而不仅仅是一个问题列表。

定义模型

这里展示了一个示例。你可以使用免费的服务，比如[JsonSchema2Pojo](#)或者这个。

```

public class Comment {

    @SerializedName("url")
    @Expose
    private String url;
    @SerializedName("html_url")
    @Expose
    private String htmlUrl;

    // getters and setters
}

```

创建 Retrofit 实例

```
ApiClient apiService = ApiClient.getInstance().create(ApiInterface.class);
```

然后，使用此实例从服务器获取数据

```

Observable<List<Issue>> issueObservable = apiService.getIssues(org, repo,
    pageNumber);
issueObservable.subscribeOn(Schedulers.newThread())
    .observeOn(AndroidSchedulers.mainThread())
    .map(issues -> issues) // 获取问题并映射到问题列表
    .subscribe(new Subscriber<List<Issue>>() {

```

```

public static Retrofit getInstance() {
    if (retrofitInstance == null) {
        retrofitInstance = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .addConverterFactory(GsonConverterFactory.create())
        .build();
    }
    return retrofitInstance;
}

public static <T> T createRetrofitService(final Class<T> clazz, final String endPoint) {
    final Retrofit restAdapter = new Retrofit.Builder()
        .baseUrl(endPoint)
        .build();

    return restAdapter.create(clazz);
}

public static String getBaseUrl() {
    return BASE_URL;
}

```

public interface ApiInterface {

```

@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                         @Path("repo") String repositoryName,
                                         @Query("page") int pageNumber);}

```

Note the getRepos() is returning an Observable and not just a list of issues.

Define the models

An example for this is shown. You can use free services like [JsonSchema2Pojo](#) or this.

```

public class Comment {

    @SerializedName("url")
    @Expose
    private String url;
    @SerializedName("html_url")
    @Expose
    private String htmlUrl;

    // getters and setters
}

```

Create Retrofit instance

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Then, Use this instance to fetch data from server

```

Observable<List<Issue>> issueObservable = apiService.getIssues(org, repo,
    pageNumber);
issueObservable.subscribeOn(Schedulers.newThread())
    .observeOn(AndroidSchedulers.mainThread())
    .map(issues -> issues) // get issues and map to issues list
    .subscribe(new Subscriber<List<Issue>>() {

```

```
    @Override
    public void onCompleted() {
        Log.i(TAG, "onCompleted: 完成！");
    }

    @Override
    public void onError(Throwable e) {
        Log.e(TAG, "onError: ", e);
    }

    @Override
    public void onNext(List<Issue> issues) {
        recyclerView.setAdapter(new IssueAdapter(MainActivity.this, issues,
        apiService));
    }
});
```

现在，您已经成功使用 Retrofit 和 RxJava 从服务器获取了数据。

```
    @Override
    public void onCompleted() {
        Log.i(TAG, "onCompleted: COMPLETED!");
    }

    @Override
    public void onError(Throwable e) {
        Log.e(TAG, "onError: ", e);
    }

    @Override
    public void onNext(List<Issue> issues) {
        recyclerView.setAdapter(new IssueAdapter(MainActivity.this, issues,
        apiService));
    }
});
```

Now, you have successfully fetched data from a server using Retrofit and RxJava.

第177章：ShortcutManager（快捷方式管理器）

第177.1节：动态启动器快捷方式

```
ShortcutManager shortcutManager = getSystemService(ShortcutManager.class);

ShortcutInfo shortcut = new ShortcutInfo.Builder(this, "id1")
.setShortLabel("网站") // 快捷方式图标标签
.setLongLabel("打开网站") // 长按应用图标时显示
.setIcon(Icon.createWithResource(context, R.drawable.icon_website))
.setIntent(new Intent(Intent.ACTION_VIEW,
        Uri.parse("https://www.mysite.example.com/")))
.build();

shortcutManager.setDynamicShortcuts(Arrays.asList(shortcut));
```

我们可以通过调用以下方法轻松移除所有动态快捷方式：

```
shortcutManager.removeAllDynamicShortcuts();
```

我们可以使用以下方法更新现有的动态快捷方式

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut));
```

请注意，`setDynamicShortcuts(List)`用于重新定义整个动态快捷方式列表，`addDynamicShortcuts(List)`用于向现有的动态快捷方式列表中添加动态快捷方式

Chapter 177: ShortcutManager

Section 177.1: Dynamic Launcher Shortcuts

```
ShortcutManager shortcutManager = getSystemService(ShortcutManager.class);

ShortcutInfo shortcut = new ShortcutInfo.Builder(this, "id1")
.setShortLabel("Web site") // Shortcut Icon tab
.setLongLabel("Open the web site") // Displayed When Long Pressing On App Icon
.setIcon(Icon.createWithResource(context, R.drawable.icon_website))
.setIntent(new Intent(Intent.ACTION_VIEW,
        Uri.parse("https://www.mysite.example.com/")))
.build();

shortcutManager.setDynamicShortcuts(Arrays.asList(shortcut));
```

We can remove all dynamic shortcuts easily by calling:

```
shortcutManager.removeAllDynamicShortcuts();
```

We can update existing Dynamic Shortcuts by Using

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut));
```

Please note that `setDynamicShortcuts(List)` is used to redefine the entire list of dynamic shortcuts, `addDynamicShortcuts(List)` is used to add dynamic shortcuts to existing list of dynamic shortcuts

第178章：LruCache

第178.1节：向缓存中添加位图（资源）

要向缓存中添加资源，必须提供一个键和资源。首先确保该值尚未存在于缓存中

```
public void addResourceToMemoryCache(String key, Bitmap resource) {  
    if (memoryCache.get(key) == null)  
        memoryCache.put(key, resource);  
}
```

第178.2节：初始化缓存

Lru缓存将存储所有添加的资源（值）以便快速访问，直到达到内存限制，此时它将丢弃使用较少的资源（值）以存储新的资源。

要初始化Lru缓存，您需要提供一个最大内存值。该值取决于您的应用需求以及资源对保持流畅应用使用的重要性。例如，对于图像库，推荐值是您最大可用内存的1/8。

还要注意，Lru缓存是基于键值对工作的。在以下示例中，键是一个String，值是一个Bitmap：

```
int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);  
int cacheSize = maxMemory / 8;  
  
LruCache<String, Bitmap> = memoryCache = new LruCache<String, Bitmap>(cacheSize) {  
    protected int sizeOf(String key, Bitmap bitmap) {  
        return bitmap.getByteCount();  
    }  
};
```

第178.3节：从缓存中获取Bitmap（资源）

要从缓存中获取资源，只需传入资源的键（此示例中为String）

```
public Bitmap 从内存缓存中获取资源(String key) {  
    memoryCache.get(key);  
}
```

Chapter 178: LruCache

Section 178.1: Adding a Bitmap(Resource) to the cache

To add a resource to the cache you must provide a key and the resource. First make sure that the value is not in the cache already

```
public void addResourceToMemoryCache(String key, Bitmap resource) {  
    if (memoryCache.get(key) == null)  
        memoryCache.put(key, resource);  
}
```

Section 178.2: Initialising the cache

The Lru Cache will store all the added resources (values) for fast access until it reaches a memory limit, in which case it will drop the less used resource (value) to store the new one.

To initialise the Lru cache you need to provide a maximum memory value. This value depends on your application requirements and in how critical the resource is to keep a smooth app usage. A recommended value for an image gallery, for example, would be 1/8 of your maximum available memory.

Also note that the Lru Cache works on a key-value basis. In the following example, the key is a `String` and the value is a `Bitmap`:

```
int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);  
int cacheSize = maxMemory / 8;  
  
LruCache<String, Bitmap> = memoryCache = new LruCache<String, Bitmap>(cacheSize) {  
    protected int sizeOf(String key, Bitmap bitmap) {  
        return bitmap.getByteCount();  
    }  
};
```

Section 178.3: Getting a Bitmap(Resource) from the cache

To get a resource from the cache simply pass the key of your resource (String in this example)

```
public Bitmap getResourceFromMemoryCache(String key) {  
    memoryCache.get(key);  
}
```

第179章：Android项目的Jenkins CI设置

第179.1节：设置Jenkins用于Android的逐步方法

这是一个使用Jenkins CI为您的Android项目设置自动构建流程的逐步指南。以下步骤假设您有一台新硬件，安装了任意版本的Linux系统。同时也考虑到您可能有一台远程机器。

第一部分：在您的机器上进行初始设置

1. 通过ssh登录到您的Ubuntu机器：

```
ssh username@xxx.xxx.xxx
```

2. 在您的机器上下载一个版本的Android SDK：

```
wget https://dl.google.com/android/android-sdk\_r24.4.1-linux.tgz
```

3. 解压下载的ar文件：

```
sudo apt-get install tar  
tar -xvf android-sdk_r24.4.1-linux.tgz
```

4. 现在你需要在你的Ubuntu机器上安装Java 8，这是Nougat上进行Android构建的要求。Jenkins需要你按照以下步骤安装JDK和JRE 7：

```
sudo apt-get install python-software-properties  
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
apt-get install openjdk-8-jdk
```

5. 现在在你的Ubuntu机器上安装Jenkins：

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -  
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'  
sudo apt-get update  
sudo apt-get install jenkins
```

Chapter 179: Jenkins CI setup for Android Projects

Section 179.1: Step by step approach to set up Jenkins for Android

This is a step by step guide to set up the automated build process using Jenkins CI for your Android projects. The following steps assume that you have new hardware with just any flavor of Linux installed. It is also taken into account that you might have a remote machine.

PART I: Initial setup on your machine

1. Log in via ssh to your Ubuntu machine:

```
ssh username@xxx.xxx.xxx
```

2. Download a version of the Android SDK on your machine:

```
wget https://dl.google.com/android/android-sdk\_r24.4.1-linux.tgz
```

3. Unzip the downloaded tar file:

```
sudo apt-get install tar  
tar -xvf android-sdk_r24.4.1-linux.tgz
```

4. Now you need to install Java 8 on your Ubuntu machine, which is a requirement for Android builds on Nougat. Jenkins would require you to install JDK and JRE 7 using the steps below:

```
sudo apt-get install python-software-properties  
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
apt-get install openjdk-8-jdk
```

5. Now install Jenkins on your Ubuntu machine:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -  
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'  
sudo apt-get update  
sudo apt-get install jenkins
```

6. 下载适用于您的安卓环境的最新支持的Gradle版本：

```
 wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip  
 unzip gradle-2.14.1-all.zip
```

7. 在你的Ubuntu机器上设置Android。首先进入Android SDK文件夹中的tools文件夹
在步骤2中下载的：

```
 cd android-sdk-linux/tools // 列出可用的SDK  
 android update sdk --no-ui // 更新SDK版本  
 android list sdk -a | grep "SDK Build-tools" // 列出可用的构建工具  
 android update sdk -a -u -t 4 // 将构建工具版本更新为上条命令列出的4  
 更新Java
```

8. 在你的机器上安装Git或其他版本控制系统：

```
 sudo apt-get install git
```

9. 现在使用你的网络浏览器登录Jenkins。在地址栏输入ipAddress:8080。

10. 为了获取首次登录的密码，请检查以下对应文件（你需要有su权限才能访问此文件）：

```
 cat /var/lib/jenkins/secrets/initialAdminPassword
```

第二部分：设置 Jenkins 构建 Android 任务

1. 登录后，进入以下路径：

```
 Jenkins > 管理 Jenkins > 全局工具配置
```

2. 在此位置，添加JAVA_HOME，内容如下：

```
 名称 = JAVA_HOME  
 JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64
```

3. 还需向Git添加以下值并保存环境变量：

6. Download the latest supported Gradle version for your Android setup:

```
 wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip  
 unzip gradle-2.14.1-all.zip
```

7. Set up Android on your Ubuntu machine. First move to the *tools* folder in the Android SDK folder
downloaded in step 2:

```
 cd android-sdk-linux/tools // lists available SDK  
 android update sdk --no-ui // Updates SDK version  
 android list sdk -a | grep "SDK Build-tools" // lists available build tools  
 android update sdk -a -u -t 4 // updates build tools version to one listed as 4 by prev. cmd.  
 update java
```

8. Install Git or any other VCS on your machine:

```
 sudo apt-get install git
```

9. Now log in to Jenkins using your internet browser. Type **ipAddress:8080** into the address bar.

10. In order to receive the password for the first-time login, please check the corresponding file as follows (you
will need su permissions to access this file):

```
 cat /var/lib/jenkins/secrets/initialAdminPassword
```

PART II: Set up Jenkins to build Android Jobs

1. Once logged in, go to the following path:

```
 Jenkins > Manage Jenkins > Global Tool Configuration
```

2. At this location, add JAVA_HOME with the following entries:

```
 Name = JAVA_HOME  
 JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64
```

3. Also add the following values to Git and save the environment variables:

名称 = 默认
/usr/bin/git

4. 现在进入以下路径：

Jenkins > 管理 Jenkins > 配置

5. 在此位置，将ANDROID_HOME添加到“全局属性”中：

名称 = ANDROID_HOME
值 = /home/username/android-sdk-linux

第三部分：为您的 Android 项目创建 Jenkins 任务

1. 点击 Jenkins 主界面上的新建项目。
2. 添加项目名称和描述。
3. 在常规标签页，选择高级。然后选择使用自定义工作区：

目录 /home/user/Code/ProjectFolder

4. 在源码管理中选择Git。此示例中我使用的是Bitbucket：

仓库 URL = <https://username:password@bitbucket.org/project/projectname.git>

5. 为您的代码库选择其他行为：

构建前清理
检出到子目录。代码库的本地子目录 /home/user/Code/ProjectFolder

6. 选择您想要构建的分支：

*/master

7. 在构建标签中，选择执行 Shell 于添加构建步骤。

8. 在执行 Shell 中，添加以下命令：

Name = Default
/usr/bin/git

4. Now go to the following path:

Jenkins > Manage Jenkins > Configuration

5. At this location, add ANDROID_HOME to the "global properties":

Name = ANDROID_HOME
Value = /home/username/android-sdk-linux

Part III: Create a Jenkins Job for your Android project

1. Click on *New Item* in the Jenkins home screen.
2. Add a *Project Name* and *Description*.
3. In the *General* tab, select *Advanced*. Then select *Use custom workspace*:

Directory /home/user/Code/ProjectFolder

4. In the source code management select *Git*. I am using *Bitbucket* for the purpose of this example:

Repository URL = <https://username:password@bitbucket.org/project/projectname.git>

5. Select additional behaviors for your repository:

Clean Before Checkout
Checkout to a sub-directory. Local subdirectory for repo /home/user/Code/ProjectFolder

6. Select a branch you want to build:

*/master

7. In the *Build* tab, select *Execute Shell* in *Add build step*.

8. In the *Execute shell*, add the following command:

```
cd /home/user/Code/ProjectFolder && gradle clean assemble --no-daemon
```

9. 如果您想对项目运行 Lint，则在执行 Shell 中添加另一个构建步骤：

```
/home/user/gradle/gradle-2.14.1/bin/gradle lint
```

现在您的系统终于设置好，可以使用 Jenkins 构建 Android 项目了。这个设置让您发布构建给 QA 和 UAT 团队的工作变得轻松许多。

附注：由于 Jenkins 是您 Ubuntu 机器上的另一个用户，您需要通过执行以下命令赋予其在工作区创建文件夹的权限：

```
chown -R jenkins .git
```

```
cd /home/user/Code/ProjectFolder && gradle clean assemble --no-daemon
```

9. If you want to run Lint on the project, then add another build step into the *Execute shell*:

```
/home/user/gradle/gradle-2.14.1/bin/gradle lint
```

Now your system is finally set up to build Android projects using Jenkins. This setup makes your life so much easier for releasing builds to QA and UAT teams.

PS: Since Jenkins is a different user on your Ubuntu machine, you should give it rights to create folders in your workspace by executing the following command:

```
chown -R jenkins .git
```

第180章：fastlane

第180.1节：用于构建并安装指定构建类型所有风味到设备的 Fastfile lane

将此 lane 添加到您的**Fastfile**中，并在命令行运行`fastlane installAll type:{BUILD_TYPE}`。将 `BUILD_TYPE` 替换为您想要构建的构建类型。

例如：`fastlane installAll type:Debug`

该命令将构建指定类型的所有风味并安装到您的设备上。目前，如果连接了多个设备，该命令无法正常工作。请确保只连接一个设备。未来我计划添加选择目标设备的选项。

```
lane :installAll do |options|
  gradle(task: "clean")
  gradle(task: "assemble",
    build_type: options[:type])
  lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do |apk|
    puts "正在上传 APK 到设备: " + apk
    begin
      adb("install -r #{apk}")
    rescue > ex
      puts ex
    end
  end
end
```

第180.2节：使用 Fastfile 构建并上传多个版本到 Crashlytics 的 Beta

这是一个多版本应用的示例Fastfile配置。它为您提供构建和部署所有版本或单个版本的选项。部署完成后，会向Slack报告部署状态，并向Beta中的Crashlytics测试者组发送通知。

要构建并部署所有版本，请使用：

```
fastlane android beta
```

要构建单个 APK 并部署，请使用：

```
fastlane android beta app:flavorName
```

使用单个 Fastlane 文件，您可以管理 iOS、Android 和 Mac 应用。如果您仅使用此文件管理一个应用，`platform` 参数不是必需的。

工作原理

Chapter 180: fastlane

Section 180.1: Fastfile lane to build and install all flavors for given build type to a device

Add this lane to your **Fastfile** and run `fastlane installAll type:{BUILD_TYPE}` in command line. Replace `BUILD_TYPE` with the build type you want to build.

For example: `fastlane installAll type:Debug`

This command will build all flavors of given type and install it to your device. Currently, it doesn't work if you have more than one device attached. Make sure you have only one. In the future I'm planning to add option to select target device.

```
lane :installAll do |options|
  gradle(task: "clean")
  gradle(task: "assemble",
    build_type: options[:type])
  lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do |apk|
    puts "Uploading APK to Device: " + apk
    begin
      adb(
        command: "install -r #{apk}"
      )
    rescue > ex
      puts ex
    end
  end
end
```

Section 180.2: Fastfile to build and upload multiple flavors to Beta by Crashlytics

This is a sample **Fastfile** setup for a multi-flavor app. It gives you an option to build and deploy all flavors or a single flavor. After the deployment, it reports to **Slack** the status of the deployment, and sends a notification to testers in Beta by Crashlytics testers group.

To build and deploy all flavors use:

```
fastlane android beta
```

To build a single APK and deploy use:

```
fastlane android beta app:flavorName
```

Using a single Fastlane file, you can manage iOS, Android, and Mac apps. If you are using this file just for one app `platform` is not required.

How It Works

1. android 参数告诉 fastlane 我们将使用:android 平台。
2. 在:android 平台内，您可以有多个 lane。目前，我只有一个:beta lane。第二个上面命令中的参数指定了我们想要使用的通道。
3. options[:app]
4. 有两个Gradle任务。首先，它运行gradle clean。如果你提供了带有app键的风味，fastfile会运行gradle assembleReleaseFlavor。否则，它运行gradle assembleRelease来构建所有构建风味。
- 5.如果我们为所有风味构建，生成的APK文件名数组会存储在 SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS中。我们使用它来循环遍历生成的文件并将它们部署到Beta by Crashlytics。notifications和groups字段是可选的。它们用于通知在Beta by Crashlytics上注册的测试人员。
- 6.如果你熟悉Crashlytics，你可能知道要在门户中激活应用，必须先在设备上运行并使用它。否则，Crashlytics会认为应用未激活并抛出错误。在这种情况下，我会捕获该错误并将其报告给Slack作为失败，这样你就会知道哪个应用未激活。
7. 如果部署成功，fastlane会向Slack发送成功消息。
8. #{/(^|/)*\$/.match(apk)} 这个正则表达式用于从APK路径中获取风味名称。如果不适合你。
9. get_version_name 和 get_version_code 是两个 Fastlane 插件，用于获取应用版本名称和版本号。如果你想使用它们，必须安装这些 gem，或者你也可以将它们移除。关于插件的更多信息请阅读[这里](#)。
10. 如果你正在构建和部署单个 APK，else 语句将会被执行。我们不需要提供 apk_path 给 Crashlytics，因为我们只有一个应用。
11. 最后的 error do 块用于在执行过程中出现其他错误时接收通知。

注意

别忘了将 SLACK_URL、API_TOKEN、GROUP_NAME 和 BUILD_SECRET 替换为自己的凭证。

```
fastlane_version "1.46.1"

default_platform :android

platform :android do

before_all do
ENV[ "SLACK_URL" ] = "https://hooks.slack.com/service...."
end

lane :beta do |options|
# 清理并构建应用的发布版本。
# 用法 `fastlane android beta app:flavorName`'

gradle(task: "clean")

gradle(task: "assemble",
build_type: "Release",
flavor: options[:app])

# 如果用户调用 `fastlane android beta` 命令，将构建所有项目并推送到 Crashlytics
if options[:app].nil?
lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |
puts "正在上传 APK 到 Crashlytics: " + apk
end
end
end
```

开始

```
crashlytics(
    api_token: "[API_TOKEN]",
    build_secret: "[BUILD_SECRET]",
```

1. android argument tells fastlane that we will use :android platform.
2. Inside :android platform you can have multiple lanes. Currently, I have only :beta lane. The second argument from the command above specifies the lane we want to use.
3. options[:app]
4. There are two **Gradle** tasks. First, it runs gradle clean. If you provided a flavor with app key, fastfile runs gradle assembleReleaseFlavor. Otherwise, it runs gradle assembleRelease to build all build flavors.
5. If we are building for all flavors, an array of generated APK file names is stored inside SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS. We use this to loop through generated files and deploy them to **Beta by Crashlytics**. notifications and groups fields are optional. They are used to notify testers registered for the app on **Beta by Crashlytics**.
6. If you are familiar with Crashlytics, you might know that to activate an app in the portal, you have to run it on a device and use it first. Otherwise, Crashlytics will assume the app inactive and throw an error. In this scenario, I capture it and report to **Slack** as a failure, so you will know which app is inactive.
7. If deployment is successful, **fastlane** will send a success message to **Slack**.
8. #{/(^|/)*\$/.match(apk)} this regex is used to get flavor name from APK path. You can remove it if it does not work for you.
9. get_version_name and get_version_code are two **Fastlane** plugins to retrieve app version name and code. You have to install these gems if you want to use, or you can remove them. Read more about Plugins [here](#).
10. The **else** statement will be executed if you are building and deploying a single APK. We don't have to provide apk_path to Crashlytics since we have only one app.
11. **error do** block at the end is used to get notified if anything else goes wrong during execution.

Note

Don't forget to replace SLACK_URL, API_TOKEN, GROUP_NAME and BUILD_SECRET with your own credentials.

```
fastlane_version "1.46.1"

default_platform :android

platform :android do

before_all do
ENV[ "SLACK_URL" ] = "https://hooks.slack.com/service...."
end

lane :beta do |options|
# Clean and build the Release version of the app.
# Usage `fastlane android beta app:flavorName`'

gradle(task: "clean")

gradle(task: "assemble",
build_type: "Release",
flavor: options[:app])

# If user calls `fastlane android beta` command, it will build all projects and push them to Crashlytics
if options[:app].nil?
lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |
puts "Uploading APK to Crashlytics: " + apk
end
end
end

begin
crashlytics(
    api_token: "[API_TOKEN]",
    build_secret: "[BUILD_SECRET]",
```

```

groups: "[GROUP_NAME]",
          apk_path: apk,
notifications: "true"
      )

slack(
message: "成功部署了新版本 #{/([^\/*$/.match(apk)}
#{get_version_name} - #{get_version_code}",
success: true,
default_payloads: [:git_branch, :lane, :test_result]
)
rescue => ex
# 如果应用在 Crashlytics 中处于非激活状态，部署将失败。在这里处理
并报告给 slack
slack(
message: "上传错误 => #{/([^\/*$/.match(apk)} #{get_version_name}
- #{get_version_code}: #{ex}",
success: false,
default_payloads: [:git_branch, :lane, :test_result]
)
end
end

after_all do |lane|
# 此代码块仅在执行的lane成功时调用
slack(
message: "操作完成，针对
#{lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].size} 个应用，版本 #{get_version_name} -
#{get_version_code}",
default_payloads: [:git_branch, :lane, :test_result],
success: true
)
end
else
# 通过Crashlytics上传单个APK到Beta
crashlytics(
api_token: "[API_TOKEN]",
build_secret: "[BUILD_SECRET]",
groups: "[GROUP_NAME]",
notifications: "true"
)

after_all do |lane|
# 此代码块仅在执行的lane成功时调用
slack(
message: "成功部署了 #{options[:app]} 的新版本
#{get_version_name} - #{get_version_code}",
default_payloads: [:git_branch, :lane, :test_result],
success: true
)
end

error do |lane, exception|
slack(
message: exception.message,
success: false,
default_payloads: [:git_branch, :lane, :test_result]
)
end
end

```

```

groups: "[GROUP_NAME]",
apk_path: apk,
notifications: "true"
)

slack(
message: "Successfully deployed new build for #{/([^\/*$/.match(apk)}
#{get_version_name} - #{get_version_code}",
success: true,
default_payloads: [:git_branch, :lane, :test_result]
)
rescue => ex
# If the app is inactive in Crashlytics, deployment will fail. Handle it here
and report to slack
slack(
message: "Error uploading => #{/([^\/*$/.match(apk)} #{get_version_name}
- #{get_version_code}: #{ex}",
success: false,
default_payloads: [:git_branch, :lane, :test_result]
)
end
end

after_all do |lane|
# This block is called, only if the executed lane was successful
slack(
message: "Operation completed for
#{lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].size} app(s) for #{get_version_name} -
#{get_version_code}",
default_payloads: [:git_branch, :lane, :test_result],
success: true
)
end
else
# Single APK upload to Beta by Crashlytics
crashlytics(
api_token: "[API_TOKEN]",
build_secret: "[BUILD_SECRET]",
groups: "[GROUP_NAME]",
notifications: "true"
)

after_all do |lane|
# This block is called, only if the executed lane was successful
slack(
message: "Successfully deployed new build for #{options[:app]} -
#{get_version_name} - #{get_version_code}",
default_payloads: [:git_branch, :lane, :test_result],
success: true
)
end

error do |lane, exception|
slack(
message: exception.message,
success: false,
default_payloads: [:git_branch, :lane, :test_result]
)
end
end

```

第181章：为自定义范围选择条（RangeSeekBar）定义步长值（增量）

Alex Florescu 在 <https://github.com/anothem/android-range-seek-bar> 提出的 Android RangeSeekBar 定制版

它允许在移动滑动条时定义步进值（增量）

第181.1节：定义步进值为7

```
<RangeSeekBar
    android:id="@+id/barPrice"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    app:barHeight="0.2dp"
    app:barHeight2="4dp"
    app:increment="7"
    app:showLabels="false" />
```

Chapter 181: Define step value (increment) for custom RangeSeekBar

A customization of the Android RangeSeekBar proposed by Alex Florescu at <https://github.com/anothem/android-range-seek-bar>

It allows to define a step value (increment), when moving the seek bar

Section 181.1: Define a step value of 7

```
<RangeSeekBar
    android:id="@+id/barPrice"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    app:barHeight="0.2dp"
    app:barHeight2="4dp"
    app:increment="7"
    app:showLabels="false" />
```

第182章：OpenGL ES 2.0+ 入门

本主题介绍在 Android 上设置和使用OpenGL ES 2.0+。OpenGL ES 是嵌入式系统上二维和三维加速图形的标准——包括游戏机、智能手机、家电和车辆。

第182.1节：设置 GLSurfaceView 和 OpenGL ES 2.0+

要在您的应用程序中使用 OpenGL ES，您必须将以下内容添加到清单文件中：

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

创建您的扩展GLSurfaceView：

```
import static android.opengl.GLES20.*; // 静态使用所有OpenGL ES 2.0的方法和常量

public class MyGLSurfaceView extends GLSurfaceView {

    public MyGLSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);

        setEGLContextClientVersion(2); // OpenGL ES 版本 2.0
        setRenderer(new MyRenderer());
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }

    public final class MyRenderer implements GLSurfaceView.Renderer{
        public final void onSurfaceCreated(GL10 unused, EGLConfig config) {
            // 您的OpenGL ES初始化方法
            glClearColor(1f, 0f, 0f, 1f);
        }
        public final void onSurfaceChanged(GL10 unused, int width, int height) {
            glViewport(0, 0, width, height);
        }

        public final void onDrawFrame(GL10 unused) {
            // 你的 OpenGL ES 绘制方法
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        }
    }
}
```

将 MyGLSurfaceView 添加到你的布局中：

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

要使用更新版本的 OpenGL ES，只需更改清单文件中的版本号、静态导入以及更改 setEGLContextClientVersion。

第182.2节：从

Chapter 182: Getting started with OpenGL ES 2.0+

This topic is about setting up and using **OpenGL ES 2.0+** on Android. OpenGL ES is the standard for 2D and 3D accelerated graphics on embedded systems - including consoles, smartphones, appliances and vehicles.

Section 182.1: Setting up GLSurfaceView and OpenGL ES 2.0+

To use OpenGL ES in your application you must add this to the manifest:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Create your extended GLSurfaceView:

```
import static android.opengl.GLES20.*; // To use all OpenGL ES 2.0 methods and constants statically

public class MyGLSurfaceView extends GLSurfaceView {

    public MyGLSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);

        setEGLContextClientVersion(2); // OpenGL ES version 2.0
        setRenderer(new MyRenderer());
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }

    public final class MyRenderer implements GLSurfaceView.Renderer{
        public final void onSurfaceCreated(GL10 unused, EGLConfig config) {
            // Your OpenGL ES init methods
            glClearColor(1f, 0f, 0f, 1f);
        }
        public final void onSurfaceChanged(GL10 unused, int width, int height) {
            glViewport(0, 0, width, height);
        }

        public final void onDrawFrame(GL10 unused) {
            // Your OpenGL ES draw methods
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        }
    }
}
```

Add MyGLSurfaceView to your layout:

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

To use newer version of OpenGL ES just change the version number in your manifest, in the static import and change setEGLContextClientVersion.

Section 182.2: Compiling and Linking GLSL-ES Shaders from

资产文件

Assets 文件夹是存放 GLSL-ES 着色器文件最常见的位置。要在你的 OpenGL ES 应用程序中使用它们，首先需要将它们加载为字符串。此函数可从资产文件创建字符串：

```
private String loadStringFromAssetFile(Context myContext, String filePath){  
    StringBuilder shaderSource = new StringBuilder();  
    try {  
        BufferedReader reader = new BufferedReader(new  
InputStreamReader(myContext.getAssets().open(filePath)));  
        String line;  
        while((line = reader.readLine()) != null){  
            shaderSource.append(line).append("");}  
  
        reader.close();  
        return shaderSource.toString();  
    } catch (IOException e) {  
        e.printStackTrace();  
        Log.e(TAG, "Could not load shader file");  
        return null;  
    }  
}
```

现在你需要创建一个函数，用于编译存储在字符串中的着色器：

```
private int compileShader(int shader_type, String shaderString){  
  
    // 这段代码从字符串编译着色器  
    int shader = glCreateShader(shader_type);  
    glShaderSource(shader, shaderString);  
    glCompileShader(shader);  
  
    // 这段代码检查编译错误  
    int[] compiled = new int[1];  
    glGetShaderiv(shader, GL_COMPILE_STATUS, compiled, 0);  
    if (compiled[0] == 0) {  
        String log = glGetShaderInfoLog(shader);  
  
        Log.e(TAG, "Shader compilation error: ");  
        Log.e(TAG, log);  
    }  
    return shader;  
}
```

现在你可以加载、编译并链接你的着色器：

```
// 从文件加载着色器  
String vertexShaderString = loadStringFromAssetFile(context, "your_vertex_shader.gls1");  
String fragmentShaderString = loadStringFromAssetFile(context, "your_fragment_shader.gls1");  
  
// 编译着色器  
int vertexShader = compileShader(GL_VERTEX_SHADER, vertexShaderString);  
int fragmentShader = compileShader(GL_FRAGMENT_SHADER, fragmentShaderString);  
  
// 连接着色器并创建着色器程序  
int shaderProgram = glCreateProgram();  
glAttachShader(shaderProgram, vertexShader);  
glAttachShader(shaderProgram, fragmentShader);  
glLinkProgram(shaderProgram);
```

asset file

The Assets folder is the most common place to store your GLSL-ES shader files. To use them in your OpenGL ES application you need to load them to a string in the first place. This functions creates a string from the asset file:

```
private String loadStringFromAssetFile(Context myContext, String filePath){  
    StringBuilder shaderSource = new StringBuilder();  
    try {  
        BufferedReader reader = new BufferedReader(new  
InputStreamReader(myContext.getAssets().open(filePath)));  
        String line;  
        while((line = reader.readLine()) != null){  
            shaderSource.append(line).append("\n");  
        }  
        reader.close();  
        return shaderSource.toString();  
    } catch (IOException e) {  
        e.printStackTrace();  
        Log.e(TAG, "Could not load shader file");  
        return null;  
    }  
}
```

Now you need to create a function that compiles a shader stored in a sting:

```
private int compileShader(int shader_type, String shaderString){  
  
    // This compiles the shader from the string  
    int shader = glCreateShader(shader_type);  
    glShaderSource(shader, shaderString);  
    glCompileShader(shader);  
  
    // This checks for for compilation errors  
    int[] compiled = new int[1];  
    glGetShaderiv(shader, GL_COMPILE_STATUS, compiled, 0);  
    if (compiled[0] == 0) {  
        String log = glGetShaderInfoLog(shader);  
  
        Log.e(TAG, "Shader compilation error: ");  
        Log.e(TAG, log);  
    }  
    return shader;  
}
```

Now you can load, compile and link your shaders:

```
// Load shaders from file  
String vertexShaderString = loadStringFromAssetFile(context, "your_vertex_shader.gls1");  
String fragmentShaderString = loadStringFromAssetFile(context, "your_fragment_shader.gls1");  
  
// Compile shaders  
int vertexShader = compileShader(GL_VERTEX_SHADER, vertexShaderString);  
int fragmentShader = compileShader(GL_FRAGMENT_SHADER, fragmentShaderString);  
  
// Link shaders and create shader program  
int shaderProgram = glCreateProgram();  
glAttachShader(shaderProgram, vertexShader);  
glAttachShader(shaderProgram, fragmentShader);  
glLinkProgram(shaderProgram);
```

```
// 检查链接错误：  
int linkStatus[] = new int[1];  
glGetProgramiv(shaderProgram, GL_LINK_STATUS, linkStatus, 0);  
if (linkStatus[0] != GL_TRUE) {  
    String log = glGetProgramInfoLog(shaderProgram);  
  
    Log.e(TAG, "无法链接着色器程序: ");  
    Log.e(TAG, log);  
}
```

如果没有错误，您的着色器程序即可使用：

```
glUseProgram(shaderProgram);
```

```
// Check for linking errors:  
int linkStatus[] = new int[1];  
glGetProgramiv(shaderProgram, GL_LINK_STATUS, linkStatus, 0);  
if (linkStatus[0] != GL_TRUE) {  
    String log = glGetProgramInfoLog(shaderProgram);  
  
    Log.e(TAG, "Could not link shader program: ");  
    Log.e(TAG, log);  
}
```

If there are no errors, your shader program is ready to use:

```
glUseProgram(shaderProgram);
```

第183章：检查数据连接

第183.1节：检查数据连接

此方法通过ping某个IP或域名来检查数据连接。

```
public Boolean isDataConnected() {
    try {
        Process p1 = java.lang.Runtime.getRuntime().exec("ping -c 1 8.8.8.8");
        int returnVal = p1.waitFor();
        boolean reachable = (returnVal==0);
        return reachable;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return false;
}
```

第183.2节：使用ConnectivityManager检查连接

```
public static boolean isConnectedNetwork (Context context) {

    ConnectivityManager cm = (ConnectivityManager)
    context.getSystemService(Context.CONNECTIVITY_SERVICE);
    return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
    ().isConnectedOrConnecting ();
}
```

第183.3节：使用网络意图在允许数据传输时执行任务

当您的设备连接到网络时，会发送一个意图。许多应用程序不会检查这些意图，但为了使您的应用程序正常工作，您可以监听网络变化意图，这些意图会告诉您何时可以进行通信。要检查网络连接，您可以例如使用以下语句：

```
if (intent.getAction().equals(android.net.ConnectivityManager.CONNECTIVITY_ACTION)){
    NetworkInfo info = intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
    //当连接到网络时执行您的操作
}
```

Chapter 183: Check Data Connection

Section 183.1: Check data connection

This method is to check data connection by ping certain IP or Domain name.

```
public Boolean isDataConnected() {
    try {
        Process p1 = java.lang.Runtime.getRuntime().exec("ping -c 1 8.8.8.8");
        int returnVal = p1.waitFor();
        boolean reachable = (returnVal==0);
        return reachable;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return false;
}
```

Section 183.2: Check connection using ConnectivityManager

```
public static boolean isConnectedNetwork (Context context) {

    ConnectivityManager cm = (ConnectivityManager)
    context.getSystemService(Context.CONNECTIVITY_SERVICE);
    return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
    ().isConnectedOrConnecting ();
}
```

Section 183.3: Use network intents to perform tasks while data is allowed

When your device connects to a network, an intent is sent. Many apps don't check for these intents, but to make your application work properly, you can listen to network change intents that will tell you when communication is possible. To check for network connectivity you can, for example, use the following clause:

```
if (intent.getAction().equals(android.net.ConnectivityManager.CONNECTIVITY_ACTION)){
    NetworkInfo info = intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
    //perform your action when connected to a network
}
```

第184章：Android上的Java

Android支持所有Java 7语言特性以及根据平台版本不同而有所差异的Java 8语言特性子集。本页介绍您可以使用的新语言特性，如何正确配置项目以使用它们，以及您可能遇到的已知问题。

第184.1节：使用Retrolambda的Java 8特性子集

[Retrolambda](#) 允许您在Java 7、6或5上运行带有lambda表达式、方法引用和try-with-resources语句的Java 8代码。它通过转换您编译后的Java 8字节码，使其能够在较旧的Java运行时环境中运行。

回移植的语言特性：

- 通过将Lambda表达式转换为匿名内部类来回移植Lambda表达式。这包括对无状态Lambda表达式使用单例实例的优化，以避免重复的对象分配。方法引用基本上只是Lambda表达式的语法糖，回移植方式相同。
- 通过移除对Throwable.addSuppressed的调用来自回移植try-with-resources语句，如果目标字节码版本低于Java 7。如果您希望被抑制的异常被记录而不是被吞掉，请创建功能请求，我们将使其可配置。
- 如果目标字节码版本低于Java 7，Objects.requireNonNull调用将被替换为Object.getClass调用。JDK 9生成的合成空检查使用Objects.requireNonNull，而早期JDK版本使用Object.getClass。
- 可选的还有：
 1. 默认方法通过将默认方法复制到伴随类（接口名 + "\$"）作为静态方法来回移植，替换接口中的默认方法为抽象方法，并向所有实现该接口的类添加必要的方法实现。
 2. 接口上的静态方法通过将静态方法移动到伴随类（接口名 + "\$"），并将所有方法调用更改为调用新方法位置来回移植。

已知限制：

- 不回移植Java 8的API
- 回移植接口上的默认方法和静态方法要求所有被回移植的接口及所有实现它们或调用其静态方法的类必须一起回移植，且只能通过一次Retrolambda执行。换句话说，您必须始终进行干净构建。此外，默认方法的回移植无法跨模块或依赖边界工作。
- 如果未来的JDK 8版本停止为每个invokedynamic调用生成新类，可能会导致故障。Retrolambda的工作原理是捕获java.lang.invoke.LambdaMetafactory动态生成的字节码，因此对该机制的优化可能会破坏Retrolambda。

[Retrolambda gradle 插件](#)将自动使用 Retrolambda 构建您的 Android 项目。最新版本可在[发布页面](#)找到。

用法：

Chapter 184: Java on Android

Android supports all Java 7 language features and a subset of Java 8 language features that vary by platform version. This page describes the new language features you can use, how to properly configure your project to use them and any known issues you may encounter.

Section 184.1: Java 8 features subset with Retrolambda

[Retrolambda](#) lets you run Java 8 code with lambda expressions, method references and try-with-resources statements on Java 7, 6 or 5. It does this by transforming your Java 8 compiled bytecode so that it can run on an older Java runtime.

Backported Language Features:

- Lambda expressions are backported by converting them to anonymous inner classes. This includes the optimisation of using a singleton instance for stateless lambda expressions to avoid repeated object allocation. Method references are basically just syntax sugar for lambda expressions and they are backported in the same way.
- Try-with-resources statements are backported by removing calls to `Throwable.addSuppressed` if the target bytecode version is below Java 7. If you would like the suppressed exceptions to be logged instead of swallowed, please create a feature request and we'll make it configurable.
- `Objects.requireNonNull` calls are replaced with calls to `Object.getClass` if the target bytecode version is below Java 7. The synthetic null checks generated by JDK 9 use `Objects.requireNonNull`, whereas earlier JDK versions used `Object.getClass`.
- Optionally also:
 1. Default methods are backported by copying the default methods to a companion class (interface name + "\$") as static methods, replacing the default methods in the interface with abstract methods, and by adding the necessary method implementations to all classes which implement that interface.
 2. Static methods on interfaces are backported by moving the static methods to a companion class (interface name + "\$"), and by changing all methods calls to call the new method location.

Known Limitations:

- Does not backport Java 8 APIs
- Backporting default methods and static methods on interfaces requires all backported interfaces and all classes which implement them or call their static methods to be backported together, with one execution of Retrolambda. In other words, you must always do a clean build. Also, backporting default methods won't work across module or dependency boundaries.
- May break if a future JDK 8 build stops generating a new class for each invokedynamic call. Retrolambda works so that it captures the bytecode that `java.lang.invoke.LambdaMetafactory` generates dynamically, so optimisations to that mechanism may break Retrolambda.

[Retrolambda gradle plugin](#) will automatically build your android project with Retrolambda. The latest version can be found on the [releases page](#).

Usage:

1. 下载并安装 jdk8
2. 在您的 build.gradle 中添加以下内容

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:<latest version>'

    }
}

// 需要此项因为 retrolambda 位于 maven central
repositories {
    mavenCentral()
}

apply plugin: 'com.android.application' //或 apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

已知问题：

- Lint 在包含 lambda 表达式的 Java 文件上会失败。Android 的 lint 不支持 Java 8 语法，可能会静默失败或报错。现在有一个实验性分支修复了该问题。
- 使用 Google Play 服务会导致 Retrolambda 失败。版本 5.0.77 包含与 Retrolambda 不兼容的字节码。这应该在较新版本的 Play 服务中得到修复，如果可以更新，建议优先采用更新版本作为解决方案。为了解决此问题，您可以使用较早的版本，如 4.4.52，或者在 JVM 参数中添加-noverify。

```
retrolambda {
    jvmArgs '-noverify'
}
```

1. Download and install [jdk8](#)
2. Add the following to your build.gradle

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:<latest version>'

    }
}

// Required because retrolambda is on maven central
repositories {
    mavenCentral()
}

apply plugin: 'com.android.application' //or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Known Issues:

- Lint fails on java files that have lambdas. Android's lint doesn't understand java 8 syntax and will fail silently or loudly. There is now an experimental fork that fixes the issue.
- Using Google Play Services causes Retrolambda to fail. Version 5.0.77 contains bytecode that is incompatible with Retrolambda. This should be fixed in newer versions of play services, if you can update, that should be the preferred solution. To work around this issue, you can either use an earlier version like 4.4.52 or add -noverify to the jvm args.

```
retrolambda {
    jvmArgs '-noverify'
}
```

第185章：Android Java本地接口 (JNI)

JNI (Java 本地接口) 是一个强大的工具，使安卓开发者能够利用 NDK 并在他们的应用中使用 C++ 本地代码。本文介绍了 Java 与 C++ 接口的使用方法。

第185.1节：如何通过JNI接口调用本地库中的函数

Java本地接口 (JNI) 允许你从Java代码调用本地函数，反之亦然。此示例展示了如何通过JNI加载并调用本地函数，但不涉及使用JNI函数访问本地代码中的Java方法和字段。

假设你有一个名为libjniexample.so的本地库，位于project/libs/<architecture>/文件夹中，并且你想调用com.example.jniexample包中JNITest Java类的一个函数。

在JNITest类中，像这样声明函数：

```
public native int testJNIfuction(int a, int b);
```

在你的本地代码中，像这样定义函数：

```
#include <jni.h>

JNIEXPORT jint JNICALL Java_com_example_jniexample_JNITest_testJNIfunction(JNIEnv *pEnv, jobject thiz, jint a, jint b)
{
    return a + b;
}
```

pEnv参数是指向JNI环境的指针，你可以将其传递给JNI函数以访问Java对象和类的方法及字段。thiz指针是调用本地方法的Java对象的jobject引用（如果是静态方法，则是类的引用）。

在你的Java代码中，在JNITest中，像这样加载库：

```
static{
    System.loadLibrary("jniexample");
}
```

注意文件名开头的lib和结尾的.so被省略了。

从Java调用本地函数的方法如下：

```
JNITest test = new JNITest();
int c = test.testJNIfunction(3, 4);
```

第185.2节：如何从本地代码调用Java方法

Java本地接口 (JNI) 允许你从本地代码调用Java函数。以下是一个简单的示例说明如何实现：

Java代码：

Chapter 185: Android Java Native Interface (JNI)

JNI (Java Native Interface) is a powerful tool that enables Android developers to utilize the NDK and use C++ native code in their applications. This topic describes the usage of Java <-> C++ interface.

Section 185.1: How to call functions in a native library via the JNI interface

The [Java Native Interface](#) (JNI) allows you to call native functions from Java code, and vice versa. This example shows how to load and call a native function via JNI, it does not go into accessing Java methods and fields from native code using [JNI functions](#).

Suppose you have a native library named libjniexample.so in the project/libs/<architecture> folder, and you want to call a function from the JNITestJava class inside the com.example.jniexample package.

In the JNITest class, declare the function like this:

```
public native int testJNIfuction(int a, int b);
```

In your native code, define the function like this:

```
#include <jni.h>

JNIEXPORT jint JNICALL Java_com_example_jniexample_JNITest_testJNIfunction(JNIEnv *pEnv, jobject thiz, jint a, jint b)
{
    return a + b;
}
```

The pEnv argument is a pointer to the JNI environment that you can pass to [JNI functions](#) to access methods and fields of Java objects and classes. The thiz pointer is a jobject reference to the Java object that the native method was called on (or the class if it is a static method).

In your Java code, in JNITest, load the library like this:

```
static{
    System.loadLibrary("jniexample");
}
```

Note the lib at the start, and the .so at the end of the filename are omitted.

Call the native function from Java like this:

```
JNITest test = new JNITest();
int c = test.testJNIfunction(3, 4);
```

Section 185.2: How to call a Java method from native code

The Java Native Interface (JNI) allows you to call Java functions from native code. Here is a simple example of how to do it:

Java code:

```

package com.example.jniexample;
public class JNITest {
    public static int getAnswer(bool) {
        return 42;
    }
}

```

本地代码：

```

int getTheAnswer()
{
    // 获取 JNI 环境
    JNIEnv *env = JniGetEnv();

    // 查找 Java 类 - 提供包名 ('.' 替换为 '/') 和类名
    jclass jniTestClass = env->FindClass("com/example/jniexample/JNITest");

    // 查找 Java 方法 - 在 () 内提供参数，返回值见下表说明如何编码
    jmethodID getAnswerMethod = env->GetStaticMethodID(jniTestClass, "getAnswer", "(Z)I");

    // 调用该方法
    return (int)env->CallStaticObjectMethod(jniTestClass, getAnswerMethod, (jboolean)true);
}

```

JNI 方法签名对应的 Java 类型：

JNI 签名	Java 类型
Z	布尔型
B	字节型
C	字符型
S	short
I	int
J	long
F	float
D	double
L 完全限定类名 ; 完全限定类名	
[类型	type[]

所以在我们的示例中，我们使用了 (Z)I —— 这意味着该函数接收一个布尔值并返回一个整数。

第185.3节：JNI层的实用方法

此方法有助于从C++字符串获取Java字符串。

```

jstring getJavaStringFromCPPString(JNIEnv *global_env, const char* cstring) {

    jstring nullString = global_env->NewStringUTF(NULL);

    if (!cstring) {
        return nullString;
    }

    jclass strClass = global_env->FindClass("java/lang/String");
    jmethodID ctorID = global_env->GetMethodID(strClass, "<init>",
        "([BLjava/lang/String;)V");

```

```

package com.example.jniexample;
public class JNITest {
    public static int getAnswer(bool) {
        return 42;
    }
}

```

Native code:

```

int getTheAnswer()
{
    // Get JNI environment
    JNIEnv *env = JniGetEnv();

    // Find the Java class - provide package ('.' replaced to '/') and class name
    jclass jniTestClass = env->FindClass("com/example/jniexample/JNITest");

    // Find the Java method - provide parameters inside () and return value (see table below for an
    // explanation of how to encode them)
    jmethodID getAnswerMethod = env->GetStaticMethodID(jniTestClass, "getAnswer", "(Z)I");

    // Calling the method
    return (int)env->CallStaticObjectMethod(jniTestClass, getAnswerMethod, (jboolean)true);
}

```

JNI method signature to Java type:

JNI Signature	Java Type
Z	boolean
B	byte
C	char
S	short
I	int
J	long
F	float
D	double
L fully-qualified-class ; fully-qualified-class	
[type	type[]

So for our example we used (Z)I - which means the function gets a boolean and returns an int.

Section 185.3: Utility method in JNI layer

This method will help to get the Java string from C++ string.

```

jstring getJavaStringFromCPPString(JNIEnv *global_env, const char* cstring) {

    jstring nullString = global_env->NewStringUTF(NULL);

    if (!cstring) {
        return nullString;
    }

    jclass strClass = global_env->FindClass("java/lang/String");
    jmethodID ctorID = global_env->GetMethodID(strClass, "<init>",
        "([BLjava/lang/String;)V");

```

```

jstring encoding = global_env->NewStringUTF("UTF-8");

jbyteArray bytes = global_env->NewByteArray(strlen(cstring));
global_env->SetByteArrayRegion(bytes, 0, strlen(cstring), (jbyte*) cstring);
jstring str = (jstring) global_env->NewObject(strClass, ctorID, bytes,
encoding);

global_env->DeleteLocalRef(strClass);
global_env->DeleteLocalRef(encoding);
global_env->DeleteLocalRef(bytes);

return str;
}

```

此方法将帮助您将 jbyteArray 转换为 char

```

char* 作为无符号字符数组(JNIEnv *env, jbyteArray array) {
    jsize length = env->GetArrayLength(array);
    jbyte* buffer = new jbyte[length + 1];

    env->GetByteArrayRegion(array, 0, length, buffer);
    buffer[length] = '\0';

    return (char*) buffer;
}

```

```

jstring encoding = global_env->NewStringUTF("UTF-8");

jbyteArray bytes = global_env->NewByteArray(strlen(cstring));
global_env->SetByteArrayRegion(bytes, 0, strlen(cstring), (jbyte*) cstring);
jstring str = (jstring) global_env->NewObject(strClass, ctorID, bytes,
encoding);

global_env->DeleteLocalRef(strClass);
global_env->DeleteLocalRef(encoding);
global_env->DeleteLocalRef(bytes);

return str;
}

```

This method will help you to convert jbyteArray to char

```

char* as_unsigned_char_array(JNIEnv *env, jbyteArray array) {
    jsize length = env->GetArrayLength(array);
    jbyte* buffer = new jbyte[length + 1];

    env->GetByteArrayRegion(array, 0, length, buffer);
    buffer[length] = '\0';

    return (char*) buffer;
}

```

方法	描述
IMPORTANCE_MAX	未使用
IMPORTANCE_HIGH	显示在所有地方，会发出声音并弹出
IMPORTANCE_DEFAULT	显示在所有地方，会发出声音，但不会视觉上打扰
IMPORTANCE_LOW	显示在所有地方，但不打扰
IMPORTANCE_MIN	仅显示在通知阴影区，折叠区域以下
IMPORTANCE_NONE	无重要性的通知；不会显示在通知阴影区

通知渠道使我们应用开发者能够将通知分组—渠道—用户可以一次性修改整个渠道的通知设置。该功能在Android O中引入，目前处于开发者预览阶段。

第186.1节：通知渠道

通知渠道是什么？

通知渠道使我们应用开发者能够将通知分组为渠道—channels—用户可以一次性修改整个渠道的通知设置。例如，对于每个渠道，用户可以完全屏蔽所有通知，覆盖重要级别，或允许显示通知徽章。这个新功能有助于大幅提升应用的用户体验。

创建通知渠道

```
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.ContextWrapper;
import android.graphics.Color;

public class NotificationUtils extends ContextWrapper {

    private NotificationManager mManager;
    public static final String ANDROID_CHANNEL_ID = "com.sai.ANDROID";
    public static final String IOS_CHANNEL_ID = "com.sai.IOS";
    public static final String ANDROID_CHANNEL_NAME = "ANDROID CHANNEL";
    public static final String IOS_CHANNEL_NAME = "IOS CHANNEL";

    public NotificationUtils(Context base) {
        super(base);
        createChannels();
    }

    public void createChannels() {

        // 创建安卓通知渠道
        NotificationChannel androidChannel = new NotificationChannel(ANDROID_CHANNEL_ID,
                ANDROID_CHANNEL_NAME, NotificationManager.IMPORTANCE_DEFAULT);
        // 设置是否在此渠道发布的通知应显示通知灯
        androidChannel.enableLights(true);
        // 设置是否在此渠道发布的通知应振动。
    }
}
```

Method	Description
IMPORTANCE_MAX	unused
IMPORTANCE_HIGH	shows everywhere, makes noise and peeks
IMPORTANCE_DEFAULT	shows everywhere, makes noise, but does not visually intrude
IMPORTANCE_LOW	shows everywhere, but is not intrusive
IMPORTANCE_MIN	only shows in the shade, below the fold
IMPORTANCE_NONE	a notification with no importance; does not show in the shade

Notification channels enable us app developers to group our notifications into groups—channels—with the user having the ability to modify notification settings for the entire channel at once. In Android O this feature is introduced. Right now it is available developers preview.

Section 186.1: Notification Channel

What Are Notification Channels?

Notification channels enable us app developers to group our notifications into groups—channels—with the user having the ability to modify notification settings for the entire channel at once. For example, for each channel, users can completely block all notifications, override the importance level, or allow a notification badge to be shown. This new feature helps in greatly improving the user experience of an app.

Create Notification Channels

```
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.ContextWrapper;
import android.graphics.Color;

public class NotificationUtils extends ContextWrapper {

    private NotificationManager mManager;
    public static final String ANDROID_CHANNEL_ID = "com.sai.ANDROID";
    public static final String IOS_CHANNEL_ID = "com.sai.IOS";
    public static final String ANDROID_CHANNEL_NAME = "ANDROID CHANNEL";
    public static final String IOS_CHANNEL_NAME = "IOS CHANNEL";

    public NotificationUtils(Context base) {
        super(base);
        createChannels();
    }

    public void createChannels() {

        // create android channel
        NotificationChannel androidChannel = new NotificationChannel(ANDROID_CHANNEL_ID,
                ANDROID_CHANNEL_NAME, NotificationManager.IMPORTANCE_DEFAULT);
        // Sets whether notifications posted to this channel should display notification lights
        androidChannel.enableLights(true);
        // Sets whether notification posted to this channel should vibrate.
    }
}
```

```

        androidChannel.enableVibration(true);
        // 设置此渠道发布的通知的通知灯颜色
        androidChannel.setLightColor(Color.BLUE);
        // 设置此渠道发布的通知是否显示在锁屏上
        androidChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

        getManager().createNotificationChannel(androidChannel);

        // 创建 iOS 通道
        NotificationChannel iosChannel = new NotificationChannel(IOS_CHANNEL_ID,
                IOS_CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
        iosChannel.enableLights(true);
        iosChannel.enableVibration(true);
        iosChannel.setLightColor(Color.GRAY);
        iosChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
        getManager().createNotificationChannel(iosChannel);

    }

private NotificationManager getManager() {
    if (mManager == null) {
        mManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    }
    return mManager;
}

```

在上述代码中，我们创建了两个NotificationChannel实例，传入唯一ID、频道名称，以及构造函数中的重要性级别。对于每个通知频道，我们应用了以下特性。

1. 声音
2. 灯光
3. 振动
4. 在锁屏上显示通知。

最后，我们从系统中获取了NotificationManager，然后通过调用方法createNotificationChannel()注册了我们创建的频道。

我们可以使用createNotificationChannels()一次性创建多个通知频道，传入一个包含NotificationChannel实例的Java列表。你可以通过getNotificationChannels()获取应用的所有通知频道，通过getNotificationChannel()传入频道ID参数获取特定频道。

通知频道中的重要级别

方法	描述
IMPORTANCE_MAX	未使用
IMPORTANCE_HIGH	显示在所有地方，会发出声音并弹出
IMPORTANCE_DEFAULT	在所有地方显示，发出声音，但不会视觉上打扰
IMPORTANCE_LOW	在所有地方显示，但不打扰，值为0
IMPORTANCE_MIN	仅显示在通知阴影区，折叠区域以下
IMPORTANCE_NONE	无重要性的通知；不会显示在通知阴影区

创建通知并发布到频道

我们创建了两个通知，一个使用NotificationUtils，另一个使用NotificationBuilder。

```

        androidChannel.enableVibration(true);
        // Sets the notification light color for notifications posted to this channel
        androidChannel.setLightColor(Color.BLUE);
        // Sets whether notifications posted to this channel appear on the lockscreen or not
        androidChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

        getManager().createNotificationChannel(androidChannel);

        // create ios channel
        NotificationChannel iosChannel = new NotificationChannel(IOS_CHANNEL_ID,
                IOS_CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
        iosChannel.enableLights(true);
        iosChannel.enableVibration(true);
        iosChannel.setLightColor(Color.GRAY);
        iosChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
        getManager().createNotificationChannel(iosChannel);

    }

private NotificationManager getManager() {
    if (mManager == null) {
        mManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    }
    return mManager;
}

```

In the code above, we created two instances of the NotificationChannel, passing uniqueid a channel name, and also an importance level in its constructor. For each notification channel, we applied following characteristics.

1. Sound
2. Lights
3. Vibration
4. Notification to show on lock screen.

Finally, we got the NotificationManager from the system and then registered the channel by calling the method createNotificationChannel(), passing the channel we have created.

We can create multiple notification channels all at once with createNotificationChannels(), passing a Java list of NotificationChannel instances. You can get all notification channels for an app with getNotificationChannels() and get a specific channel with getNotificationChannel(), passing only the channel id as an argument.

Importance Level in Notification Channels

Method	Description
IMPORTANCE_MAX	unused
IMPORTANCE_HIGH	shows everywhere, makes noise and peeks
IMPORTANCE_DEFAULT	shows everywhere, makes noise, but does not visually intrude
IMPORTANCE_LOW	shows everywhere, but is not intrusive,value is 0
IMPORTANCE_MIN	only shows in the shade, below the fold
IMPORTANCE_NONE	a notification with no importance; does not show in the shade

Create Notification and Post to channel

We have created two notification one using NotificationUtils another using NotificationBuilder.

```

public Notification.Builder getAndroidChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), ANDROID_CHANNEL_ID)
        .setContentTitle(title)
        .setContentText(body)
        .setSmallIcon(android.R.drawable.stat_notify_more)
        .setAutoCancel(true);
}

Builder getIosChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), IOS_CHANNEL_ID)
        .setContentTitle(title)
        .setContentText(body)
        .setSmallIcon(android.R.drawable.stat_notify_more)
        .setAutoCancel(true);
}

```

我们也可以使用 `Notification.Builder()` 设置通知渠道。为此，我们可以使用 `setChannel(String channelId)`。

更新通知渠道设置

一旦您创建了通知渠道，用户将负责其设置和行为。您可以再次调用 `createNotificationChannel()` 来重命名现有的通知渠道，或更新其描述。以下示例代码描述了如何通过创建一个意图来启动活动，将用户重定向到通知渠道的设置。在这种情况下，意图需要包括通知渠道ID和您的应用包名的扩展数据。

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    //...
    Button buttonAndroidNotifSettings = (Button) findViewById(R.id.btn_android_notif_settings);
    buttonAndroidNotifSettings.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View view) {
            Intent i = new Intent(Settings.ACTION_CHANNEL_NOTIFICATION_SETTINGS);
            i.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            i.putExtra(Settings.EXTRA_CHANNEL_ID, NotificationUtils.ANDROID_CHANNEL_ID);
            startActivity(i);
        }
    });
}

```

XML 文件：

```

<!-- . . -->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="通知设置"/>
<!-- . . -->

```

删除通知渠道

您可以通过调用 `deleteNotificationChannel()` 来删除通知渠道。

```

NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

```

```

public Notification.Builder getAndroidChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), ANDROID_CHANNEL_ID)
        .setContentTitle(title)
        .setContentText(body)
        .setSmallIcon(android.R.drawable.stat_notify_more)
        .setAutoCancel(true);
}

public Notification.Builder getIosChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), IOS_CHANNEL_ID)
        .setContentTitle(title)
        .setContentText(body)
        .setSmallIcon(android.R.drawable.stat_notify_more)
        .setAutoCancel(true);
}

```

We can also set `NotificationChannel` Using `Notification.Builder()`. For that we can use `setChannel(String channelId)`.

Update Notification Channel Settings

Once you create a notification channel, the user is in charge of its settings and behavior. You can call `createNotificationChannel()` again to rename an existing notification channel, or update its description. The following sample code describes how you can redirect a user to the settings for a notification channel by creating an intent to start an activity. In this case, the intent requires extended data including the ID of the notification channel and the package name of your app.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    //...
    Button buttonAndroidNotifSettings = (Button) findViewById(R.id.btn_android_notif_settings);
    buttonAndroidNotifSettings.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View view) {
            Intent i = new Intent(Settings.ACTION_CHANNEL_NOTIFICATION_SETTINGS);
            i.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            i.putExtra(Settings.EXTRA_CHANNEL_ID, NotificationUtils.ANDROID_CHANNEL_ID);
            startActivity(i);
        }
    });
}

```

XML file:

```

<!-- . . -->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!-- . . -->

```

Deleting Notification Channel

You can delete notification channels by calling `deleteNotificationChannel()`.

```

NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

```

```
// 频道的ID。
String id = "my_channel_01";
NotificationChannel mChannel = mNotificationManager.getNotificationChannel(id);
mNotificationManager.deleteNotificationChannel(mChannel);
```

现在创建 MainActivity 和 xml

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="16dp"
    tools:context="com.chikeandroid.tutsplusalerts.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        &lt;TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tuts+ Android Channel"
            android:layout_gravity="center_horizontal"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
            android:id="@+id/et_android_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="标题"/>

        <EditText
            android:id="@+id/et_android_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="作者"/>

        <Button
            android:id="@+id/btn_send_android"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="发送"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="20dp">

        &lt;TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tuts+ Android Channel"
            android:layout_gravity="center_horizontal"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
            android:id="@+id/et_android_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Title"/>

        <EditText
            android:id="@+id/et_android_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Author"/>

        <Button
            android:id="@+id/btn_send_android"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send"/>
    </LinearLayout>

```

```
// The id of the channel.
String id = "my_channel_01";
NotificationChannel mChannel = mNotificationManager.getNotificationChannel(id);
mNotificationManager.deleteNotificationChannel(mChannel);
```

Now Create MainActivity and xml

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="16dp"
    tools:context="com.chikeandroid.tutsplusalerts.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tuts+ Android Channel"
            android:layout_gravity="center_horizontal"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
            android:id="@+id/et_android_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="标题"/>

        <EditText
            android:id="@+id/et_android_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="作者"/>

        <Button
            android:id="@+id/btn_send_android"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="发送"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="20dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tuts+ Android Channel"
            android:layout_gravity="center_horizontal"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
            android:id="@+id/et_android_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Title"/>

        <EditText
            android:id="@+id/et_android_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Author"/>

        <Button
            android:id="@+id/btn_send_android"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send"/>
    </LinearLayout>

```

```
    android:text="Tuts+ IOS 频道"
    android:layout_gravity="center_horizontal"
    android:textAppearance="@style/TextAppearance.AppCompat.Title"/>
```

```
<EditText
    android:id="@+id/et_ios_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="标题"
    />

<EditText
    android:id="@+id/et_ios_author"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="作者"/>
<Button
    android:id="@+id	btn_send_ios"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="发送"/>
</LinearLayout>
</LinearLayout>
```

MainActivity.java

我们将编辑我们的 MainActivity，以便从 EditText 组件获取标题和作者，然后将这些发送到 Android 通道。我们获取在 NotificationUtils 中创建的 Android 通道的 Notification.Builder，然后通知 NotificationManager。

```
import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils; import android.view.View; import android.widget.Button; import
android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    private NotificationUtils mNotificationUtils;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNotificationUtils = new NotificationUtils(this);

        final EditText editTextTitleAndroid = (EditText) findViewById(R.id.et_android_title);
        final EditText editTextAuthorAndroid = (EditText) findViewById(R.id.et_android_author);
        Button buttonAndroid = (Button) findViewById(R.id.btn_send_android);

        buttonAndroid.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = editTextTitleAndroid.getText().toString();
                String author = editTextAuthorAndroid.getText().toString();

                if(!TextUtils.isEmpty(title) && !TextUtils.isEmpty(author)) {
                    Notification.Builder nb = mNotificationUtils.
                        getAndroidChannelNotification(title, "By " + author);
                }
            }
        });
    }
}
```

```
    android:text="Tuts+ IOS Channel"
    android:layout_gravity="center_horizontal"
    android:textAppearance="@style/TextAppearance.AppCompat.Title"/>
```

```
<EditText
    android:id="@+id/et_ios_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Title"
    />

<EditText
    android:id="@+id/et_ios_author"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Author"/>
<Button
    android:id="@+id	btn_send_ios"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Send"/>
</LinearLayout>
</LinearLayout>
```

MainActivity.java

we are going to edit our MainActivity so that we can get the title and author from the EditText components and then send these to the Android channel. We get the Notification.Builder for the Android channel we created in our NotificationUtils, and then notify the NotificationManager.

```
import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils; import android.view.View; import android.widget.Button; import
android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    private NotificationUtils mNotificationUtils;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNotificationUtils = new NotificationUtils(this);

        final EditText editTextTitleAndroid = (EditText) findViewById(R.id.et_android_title);
        final EditText editTextAuthorAndroid = (EditText) findViewById(R.id.et_android_author);
        Button buttonAndroid = (Button) findViewById(R.id.btn_send_android);

        buttonAndroid.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = editTextTitleAndroid.getText().toString();
                String author = editTextAuthorAndroid.getText().toString();

                if(!TextUtils.isEmpty(title) && !TextUtils.isEmpty(author)) {
                    Notification.Builder nb = mNotificationUtils.
                        getAndroidChannelNotification(title, "By " + author);
                }
            }
        });
    }
}
```

```
mNotificationUtils.getManager().notify(107, nb.build());  
    }  
}  
});  
}  
}
```

```
mNotificationUtils.getManager().notify(107, nb.build());  
    }  
}  
});  
}
```

第187章：Robolectric

单元测试是将一段代码独立测试，不依赖系统中其他任何依赖或部分（例如数据库）运行。

Robolectric 是一个单元测试框架，它去除了 Android SDK jar 的限制，使您能够测试驱动开发您的 Android 应用。测试在您的工作站上的 JVM 内几秒钟内运行完成。

将两者结合起来，可以让你在仍然使用Android API的情况下快速运行JVN测试。

第187.1节：Robolectric测试

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

    @Test
    public void 点击按钮_应改变结果视图文本() throws Exception {
        MyActivity activity = Robolectric.setupActivity(MyActivity.class);

        Button button = (Button) activity.findViewById(R.id.button);
        TextView results = (TextView) activity.findViewById(R.id.results);

        button.performClick();
        assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
    }
}
```

第187.2节：配置

要配置robolectric，请在测试类或方法上添加@Config注解。

使用自定义Application类运行

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
```

设置目标 SDK

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
```

使用自定义清单运行

指定时，robolectric 将相对于当前目录查找。默认值是AndroidManifest.xml

资源和资产将相对于清单加载。

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
```

Chapter 187: Robolectric

Unit testing is taking a piece of code and testing it independently without any other dependencies or parts of the system running (for example the database).

Robolectric is a unit test framework that de-fangs the Android SDK jar so you can test-drive the development of your Android app. Tests run inside the JVM on your workstation in seconds.

Combing them both allows you to run fast tests on the JVN still using the Android API's.

Section 187.1: Robolectric test

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

    @Test
    public void clickingButton_shouldChangeResultsViewText() throws Exception {
        MyActivity activity = Robolectric.setupActivity(MyActivity.class);

        Button button = (Button) activity.findViewById(R.id.button);
        TextView results = (TextView) activity.findViewById(R.id.results);

        button.performClick();
        assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
    }
}
```

Section 187.2: Configuration

To configure robolectric add @Config annotation to test class or method.

Run with custom Application class

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
```

Set target SDK

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
```

Run with custom manifest

When specified, robolectric will look relative to the current directory. Default value is `AndroidManifest.xml`

Resources and assets will be loaded relative to the manifest.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
```

使用限定符

可能的限定符可以在[android docs](#)中找到。

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Use qualifiers

Possible qualifiers can be found in [android docs](#).

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

第188章：Moshi

Moshi 是一个适用于 Android 和 Java 的现代 JSON 库。它使将 JSON 解析为 Java 对象以及将 Java 对象转换回 JSON 变得简单。

第188.1节：将 JSON 转换为 Java

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

第188.2节：将 Java 对象序列化为 JSON

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', 黑桃),
    Arrays.asList(new Card('4', 梅花), new Card('A', 红心)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

第188.3节：内置类型适配器

Moshi 内置支持读取和写入 Java 核心数据类型：

- 基本类型 (int、float、char 等) 及其包装类 (Integer、Float、Character 等)。
- 数组
- 集合
- 列表
- 集合
- 映射 字符串 枚举

它通过逐字段写出支持您的模型类。在上面的示例中，Moshi 使用了这些类：

```
class BlackjackHand {
    public final Card hidden_card;
    public final List<Card> visible_cards;
    ...
}

class Card {
    public final char rank;
    public final Suit suit;
    ...
}

enum Suit {
    CLUBS, DIAMONDS, HEARTS, SPADES;
}
```

用于读取和写入此 JSON：

Chapter 188: Moshi

Moshi is a modern JSON library for Android and Java. It makes it easy to parse JSON into Java objects and Java back into JSON.

Section 188.1: JSON into Java

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

Section 188.2: serialize Java objects as JSON

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', SPADES),
    Arrays.asList(new Card('4', CLUBS), new Card('A', HEARTS)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

Section 188.3: Built in Type Adapters

Moshi has built-in support for reading and writing Java's core data types:

- Primitives (int, float, char...) and their boxed counterparts (Integer, Float, Character...).
- Arrays
- Collections
- Lists
- Sets
- Maps Strings Enums

It supports your model classes by writing them out field-by-field. In the example above Moshi uses these classes:

```
class BlackjackHand {
    public final Card hidden_card;
    public final List<Card> visible_cards;
    ...
}

class Card {
    public final char rank;
    public final Suit suit;
    ...
}

enum Suit {
    CLUBS, DIAMONDS, HEARTS, SPADES;
}
```

to read and write **this** JSON:

```
{  
    "hidden_card": {  
        "rank": "6",  
        "suit": "SPADES"  
    },  
    "visible_cards": [  
        {  
            "rank": "4",  
            "suit": "梅花"  
        },  
        {  
            "rank": "A",  
            "suit": "红心"  
        }  
    ]  
}
```

```
{  
    "hidden_card": {  
        "rank": "6",  
        "suit": "SPADES"  
    },  
    "visible_cards": [  
        {  
            "rank": "4",  
            "suit": "CLUBS"  
        },  
        {  
            "rank": "A",  
            "suit": "HEARTS"  
        }  
    ]  
}
```

第189章：严格模式策略：一种在编译时捕捉错误的工具。

严格模式是Android 2.3引入的一个用于调试的特殊类。该开发者工具可以检测意外发生的操作并提醒我们，以便修复。它最常用于捕捉应用主线程上的意外磁盘或网络访问，主线程是接收UI操作和执行动画的地方。严格模式基本上是一个在编译时捕捉错误的工具。

第189.1节：以下代码片段用于设置线程策略的严格模式。此代码应在应用程序的入口点设置。

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //向LogCat记录一条消息
    .build())
```

第189.2节：以下代码处理内存泄漏，比如检测SQLite的finalize方法是否被调用。

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Chapter 189: Strict Mode Policy : A tool to catch the bug in the Compile Time.

Strict Mode is a special class introduced in Android 2.3 for debugging. This developer tools detect things done accidentally and bring them to our attention so that we can fix them. It is most commonly used to catch the accidental disk or network access on the applications' main thread, where UI operations are received and animations takes place. StrictMode is basically a tool to catch the bug in the Compile Time mode.

Section 189.1: The below Code Snippet is to setup the StrictMode for Thread Policies. This Code is to be set at the entry points to our application

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

Section 189.2: The below code deals with leaks of memory, like it detects when in SQLite finalize is called or not

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

第190章：国际化和本地化（I18N 和 L10N）

国际化（i18n）和本地化（L10n）用于根据语言差异、地区差异和目标受众调整软件。

国际化：为未来本地化做规划的过程，即使软件设计具有一定的灵活性，能够调整和适应未来的本地化工作。

本地化：将软件适应特定地区/国家/市场（区域设置）的过程。

第190.1节：本地化规划：在

Manifest中启用RTL支持

RTL（从右到左）支持是国际化和本地化规划中的重要部分。与从左到右书写的英语不同，许多语言如阿拉伯语、日语、希伯来语等是从右到左书写的。为了吸引更广泛的全球受众，最好从项目一开始就规划布局以支持这些语言，这样后续添加本地化会更容易。

可以通过在AndroidManifest中添加supportsRtl标签来启用Android应用中的RTL支持，示例如下：

```
<application
    ...
    android:supportsRtl="true"
    ...
</application>
```

第190.2节：本地化规划：在布局中添加RTL支持

从SDK 17（Android 4.2）开始，Android布局中添加了RTL支持，这是本地化的重要组成部分。

今后，布局中的left/right表示法应替换为start/end表示法。但是，如果您的项目的minSdk值小于17，则布局中应同时使用left/right和start/end表示法。

对于相对布局，应使用alignParentStart和alignParentEnd，如下所示：

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    &lt;TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>
    &lt;TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

Chapter 190: Internationalization and localization (I18N and L10N)

Internationalization (i18n) and Localization (L10n) are used to adapt software according to differences in languages, regional differences and target audience.

Internationalization : the process of planning for future localization i.e. making the software design flexible to an extent that it can adjust and adapt to future localization efforts.

Localization : the process of adapting the software to a particular region/country/market (locale).

Section 190.1: Planning for localization : enable RTL support in Manifest

RTL (Right-to-left) support is an essential part in planning for i18n and L10n. Unlike English language which is written from left to right, many languages like Arabic, Japanese, Hebrew, etc. are written from right to left. To appeal to a more global audience, it is a good idea to plan your layouts to support these language from the very beginning of the project, so that adding localization is easier later on.

RTL support can be enabled in an Android app by adding the supportsRtl tag in the AndroidManifest, like so :

```
<application
    ...
    android:supportsRtl="true"
    ...
</application>
```

Section 190.2: Planning for localization : Add RTL support in Layouts

Starting SDK 17 (Android 4.2), RTL support was added in Android layouts and is an essential part of localization. Going forward, the left/right notation in layouts should be replaced by start/end notation. If, however, your project has a minSdk value less than 17, then both left/right and start/end notation should be used in layouts.

For relative layouts, alignParentStart and alignParentEnd should be used, like so:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

指定重力和布局重力时，应使用类似的表示法，如下所示：

```
&lt;TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="left|start"  
    android:gravity="left|start"/>  
  
&lt;TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="right|end"  
    android:gravity="right|end"/>
```

内边距和外边距也应相应指定，如下所示：

```
<include layout="@layout/notification"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="12dp"  
    android:layout_marginStart="12dp"  
    android:paddingLeft="128dp"  
    android:paddingStart="128dp"  
    android:layout_toLeftOf="@+id/cancel_action"  
    android:layout_toStartOf="@+id/cancel_action"/>  
  
<include layout="@layout/notification2"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginRight="12dp"  
    android:layout_marginEnd="12dp"  
    android:paddingRight="128dp"  
    android:paddingEnd="128dp"  
    android:layout_toRightOf="@+id/cancel_action"  
    android:layout_toEndOf="@+id/cancel_action"/>
```

第190.3节：本地化规划：RTL布局测试

要测试已创建的布局是否兼容RTL，请执行以下操作：

进入设置 -> 开发者选项 -> 绘图 -> 强制RTL布局方向

启用此选项将强制设备使用RTL（从右到左）语言环境，您可以轻松验证应用的所有部分是否支持RTL。
请注意，到此为止，您实际上不需要添加任何新的语言环境/语言支持。

第190.4节：本地化编码：创建默认字符串和资源

本地化编码的第一步是创建默认资源。这一步非常隐含，许多开发者甚至不会考虑它。然而，创建默认资源非常重要，因为如果设备运行在不支持的语言环境下，它会从默认文件夹加载所有资源。如果默认文件夹中缺少任何一个资源，应用程序就会崩溃。

默认字符串集应放置在以下指定位置的文件夹中：

res/values/strings.xml

For specifying gravity and layout gravity, similar notation should be used, like so :

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="left|start"  
    android:gravity="left|start"/>  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="right|end"  
    android:gravity="right|end"/>
```

Paddings and margins should also be specified accordingly, like so :

```
<include layout="@layout/notification"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="12dp"  
    android:layout_marginStart="12dp"  
    android:paddingLeft="128dp"  
    android:paddingStart="128dp"  
    android:layout_toLeftOf="@+id/cancel_action"  
    android:layout_toStartOf="@+id/cancel_action"/>  
  
<include layout="@layout/notification2"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginRight="12dp"  
    android:layout_marginEnd="12dp"  
    android:paddingRight="128dp"  
    android:paddingEnd="128dp"  
    android:layout_toRightOf="@+id/cancel_action"  
    android:layout_toEndOf="@+id/cancel_action"/>
```

Section 190.3: Planning for localization : Test layouts for RTL

To test if the layouts that have been created are RTL compatible, do the following :

Go to Settings -> Developer options -> Drawing -> Force RTL layout direction

Enabling this option would force the device to use RTL locales and you can easily verify all parts of the app for RTL support. Note that you don't need to actually add any new locales/ language support up till this point.

Section 190.4: Coding for Localization : Creating default strings and resources

The first step for coding for localization is to create default resources. This step is so implicit that many developers do not even think about it. However, creating default resources is important because if the device runs on an unsupported locale, it would load all of its resources from the default folders. If even one of the resources is missing from the default folders, the app would simply crash.

The default set of strings should be put in the following folder at the specified location:

res/values/strings.xml

该文件应包含应用大多数用户预计使用的语言的字符串。

此外，应用的默认资源应放置在以下文件夹和位置：

res/drawable/
res/layout/

如果您的应用需要像anim或xml这样的文件夹， 默认资源应添加到以下文件夹和位置：

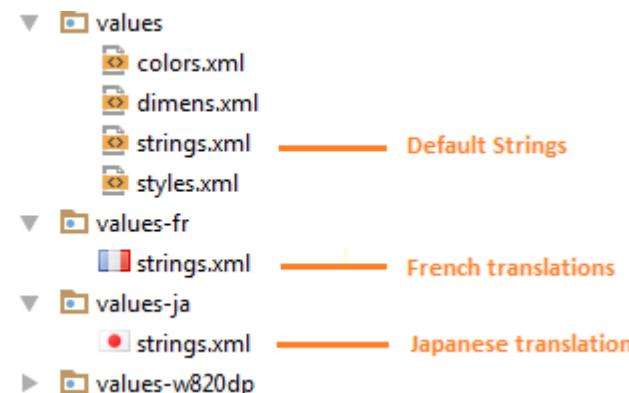
res/anim/
res/xml/
res/raw/

第190.5节：本地化编码：提供替代字符串

为了提供其他语言（地区）的翻译，我们需要按照以下约定在单独的文件夹中创建一个strings.xml文件：

res/values-<locale>/strings.xml

下面给出了一个示例：



在此示例中， 默认的英文字符串位于文件res/values/strings.xml中， 法语翻译位于文件夹res/values-fr/strings.xml中， 日语翻译位于文件夹res/values-ja/strings.xml

其他地区的翻译也可以类似地添加到应用中。

完整的地区代码列表可在此处找到：[ISO 639 codes](#)

不可翻译的字符串：

您的项目中可能有某些字符串不需要翻译。作为SharedPreferences键使用的字符串或作为符号使用的字符串属于此类。这些字符串应仅存储在默认的strings.xml中，并应标记为translatable="false"属性。例如：

```
<string name="pref_widget_display_label_hot">热点新闻</string>
<string name="pref_widget_display_key" translatable="false">widget_display</string>
<string name="pref_widget_display_hot" translatable="false">0</string>
```

This file should contain the strings in the language that majority users of the app are expected to speak.

Also, default resources for the app should be placed at the following folders and locations :

res/drawable/
res/layout/

If your app requires folders like anim, or xml, the default resources should be added to the following folders and locations:

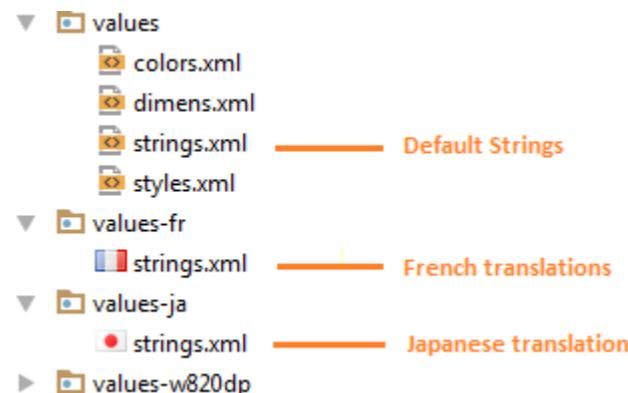
res/anim/
res/xml/
res/raw/

Section 190.5: Coding for localization : Providing alternative strings

To provide translations in other languages (locales), we need to create a `strings.xml` in a separate folder by the following convention :

res/values-<locale>/strings.xml

An example for the same is given below:



In this example, we have default English strings in the file `res/values/strings.xml`, French translations are provided in the folder `res/values-fr/strings.xml` and Japanese translations are provided in the folder `res/values-ja/strings.xml`

Other translations for other locales can similarly be added to the app.

A complete list of locale codes can be found here : [ISO 639 codes](#)

Non-translatable Strings:

Your project may have certain strings that are not to be translated. Strings which are used as keys for SharedPreferences or strings which are used as symbols, fall in this category. These strings should be stored only in the default `strings.xml` and should be marked with a `translatable="false"` attribute. e.g.

```
<string name="pref_widget_display_label_hot">Hot News</string>
<string name="pref_widget_display_key" translatable="false">widget_display</string>
<string name="pref_widget_display_hot" translatable="false">0</string>
```

该属性很重要，因为翻译通常由双语专业人员进行。这使得参与翻译的人能够识别不需要翻译的字符串，从而节省时间和金钱。

第190.6节：本地化编码：提供备用布局

如果您已经指定了正确的start/end标记，如前面的示例所述，通常不需要创建特定语言的布局。然而，有些情况下默认布局可能无法正确适用于某些语言。有时，左到右的布局无法适用于从右到左的语言。在这种情况下，必须提供正确的布局。

为了对从右到左（RTL）布局进行完全优化，我们可以使用完全独立的布局文件，使用ldrtl资源限定符（ldrtl代表layout-direction-right-to-left）。例如，我们可以将默认布局文件保存在res/layout/目录下，将RTL优化的布局保存在res/layout-ldrtl/目录下。

ldrtl限定符非常适合用于drawable资源，这样您可以提供与阅读方向相对应的图形。

这里有一篇很好的文章，描述了ldrtl布局的优先级：语言特定布局

This attribute is important because translations are often carried out by professionals who are bilingual. This would allow these persons involved in translations to identify strings which are not to be translated, thus saving time and money.

Section 190.6: Coding for localization : Providing alternate layouts

Creating language specific layouts is often unnecessary if you have specified the correct start/end notation, as described in the earlier example. However, there may be situations where the default layouts may not work correctly for certain languages. Sometimes, left-to-right layouts may not translate for RTL languages. It is necessary to provide the correct layouts in such cases.

To provide complete optimization for RTL layouts, we can use entirely separate layout files using the ldrtl resource qualifier (ldrtl stands for layout-direction-right-to-left). For example, we can save your default layout files in res/layout/ and our RTL optimized layouts in res/layout-ldrtl/.

The ldrtl qualifier is great for drawable resources, so that you can provide graphics that are oriented in the direction corresponding to the reading direction.

Here is a great post which describes the precedence of the ldrtl layouts : [Language specific layouts](#)

第191章：在安卓项目中快速设置Retrolambda的方法。

Retrolambda 是一个库，允许在 Java 7、6 或 5 上使用 Java 8 的 lambda 表达式、方法引用和 try-with-resources 语句。

Gradle Retrolambda 插件允许将 Retrolambda 集成到基于 Gradle 的构建中。例如，这允许在 Android 应用中使用这些结构，因为当前标准的 Android 开发尚不支持 Java 8。

第 191.1 节：设置及使用示例：

设置步骤：

1. 下载并安装 jdk8。
2. 将以下内容添加到项目的主 build.gradle 中

```
buildscript {  
    repositories {  
        mavenCentral()  
    }  
  
    dependencies {  
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'  
    }  
}  
}
```

3. 现在将此添加到应用模块的 build.gradle 中

```
apply plugin: 'com.android.application' // 或者 apply plugin: 'java'  
apply plugin: 'me.tatarka.retrolambda'
```

4. 将这些行添加到您的应用模块的 build.gradle 中，以通知 IDE 语言级别：

```
android {  
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
}
```

示例：

所以像这样的代码：

```
button.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        log("Clicked");  
    }  
});
```

变成这样：

Chapter 191: Fast way to setup Retrolambda on an android project.

Retrolambda is a library which allows to use Java 8 lambda expressions, method references and try-with-resources statements on Java 7, 6 or 5.

The Gradle Retrolambda Plug-in allows to integrate Retrolambda into a Gradle based build. This allows for example to use these constructs in an Android application, as standard Android development currently does not yet support Java 8.

Section 191.1: Setup and example how to use:

Setup Steps:

1. Download and install jdk8.
2. Add the following to your project's main build.gradle

```
buildscript {  
    repositories {  
        mavenCentral()  
    }  
  
    dependencies {  
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'  
    }  
}  
}
```

3. Now add this to your application module's build.gradle

```
apply plugin: 'com.android.application' // or apply plugin: 'java'  
apply plugin: 'me.tatarka.retrolambda'
```

4. Add these lines to your application module's build.gradle to inform the IDE of the language level:

```
android {  
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
}
```

Example:

So things like this:

```
button.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        log("Clicked");  
    }  
});
```

Become this:

```
button.setOnClickListener(v -> log("Clicked"));
```

```
button.setOnClickListener(v -> log("Clicked"));
```

第192章：如何使用SparseArray

A [SparseArray](#) 是 Map 的替代方案。Map 要求其键必须是对象。当我们想使用原始类型的 int 值作为键时，会发生自动装箱现象。编译器会自动将原始值转换为其包装类型（例如 int 转换为 Integer）。内存占用的差异是显著的：int 使用 4 字节，Integer 使用 16 字节。SparseArray 使用 int 作为键值。

第192.1节：使用SparseArray的基本示例

```
类 Person {
    字符串 name;

    公共 Person(字符串 name) {
        this.name = name;
    }

    @Override
    公共 布尔型 equals(对象 o) {
        如果 (this == o) 返回 真;
        如果 (o == 空 || getClass() != o.getClass()) 返回 假;

        Person person = (Person) o;

        返回 name != 空 ? name.equals(person.name) : person.name == 空;
    }

    @Override
    公共 整型 hashCode() {
        返回 name != 空 ? name.hashCode() : 0;
    }

    @Override
    公共 字符串 toString() {
        返回 "Person{" +
            "name='" + name + '\'';
    }
}

final Person steve = new Person("Steve");
Person[] persons = new Person[] { new Person("John"), new Person("Gwen"), steve, new Person("Rob") };
int[] identifiers = new int[] {1234, 2345, 3456, 4567};

final SparseArray<Person> demo = new SparseArray<>();

// 将人员映射到标识符。
for (int i = 0; i < persons.length; i++) {
    demo.put(identifiers[i], persons[i]);
}

// 查找标识符为1234的人。
Person id1234 = demo.get(1234); // 返回 John.

// 查找标识符为6410的人。
Person id6410 = demo.get(6410); // 返回 null.

// 查找第3个人。
Person third = demo.valueAt(3); // 返回 Rob.
```

Chapter 192: How to use SparseArray

A [SparseArray](#) is an alternative for a [Map](#). A [Map](#) requires its keys to be objects. The phenomenon of autoboxing occurs when we want to use a primitive [int](#) value as key. The compiler automatically converts primitive values to their boxed types (e.g. [int](#) to [Integer](#)). The difference in memory footprint is noticeable: [int](#) uses 4 bytes, [Integer](#) uses 16 bytes. A [SparseArray](#) uses [int](#) as key value.

Section 192.1: Basic example using SparseArray

```
class Person {
    String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        return name != null ? name.equals(person.name) : person.name == null;
    }

    @Override
    public int hashCode() {
        return name != null ? name.hashCode() : 0;
    }

    @Override
    public String toString() {
        return "Person{" +
            "name='" + name + '\'';
    }
}

final Person steve = new Person("Steve");
Person[] persons = new Person[] { new Person("John"), new Person("Gwen"), steve, new Person("Rob") };
int[] identifiers = new int[] {1234, 2345, 3456, 4567};

final SparseArray<Person> demo = new SparseArray<>();

// Mapping persons to identifiers.
for (int i = 0; i < persons.length; i++) {
    demo.put(identifiers[i], persons[i]);
}

// Find the person with identifier 1234.
Person id1234 = demo.get(1234); // Returns John.

// Find the person with identifier 6410.
Person id6410 = demo.get(6410); // Returns null.

// Find the 3rd person.
Person third = demo.valueAt(3); // Returns Rob.
```

```
// 查找第42个人。  
//Person fortysecond = demo.getValueAt(42); // 抛出 ArrayIndexOutOfBoundsException.  
  
// 删除最后一个人。  
demo.removeAt(demo.size() - 1); // Rob 被删除。  
  
// 删除标识符为1234的人。  
demo.delete(1234); // John 被删除。  
  
// 查找 Steve 的索引。  
int indexOfSteve = demo.indexOfValue(StringUtils.fromString("Steve"));  
  
// 查找 Steve 的标识符。  
int identifierOfSteve = demo.keyAt(indexOfSteve);
```

[YouTube教程](#)

```
// Find the 42th person.  
//Person fortysecond = demo.getValueAt(42); // Throws ArrayIndexOutOfBoundsException.  
  
// Remove the last person.  
demo.removeAt(demo.size() - 1); // Rob removed.  
  
// Remove the person with identifier 1234.  
demo.delete(1234); // John removed.  
  
// Find the index of Steve.  
int indexOfSteve = demo.indexOfValue(StringUtils.fromString("Steve"));  
  
// Find the identifier of Steve.  
int identifierOfSteve = demo.keyAt(indexOfSteve);
```

[Tutorial on YouTube](#)

第193章：共享元素过渡

这里提供了使用共享元素在Activities或Fragments之间进行过渡的示例。一个此类行为的例子是Google Play商店应用，它将应用图标从列表视图过渡到应用详情视图。

第193.1节：两个

Fragments之间的共享元素过渡

在此示例中，两个不同的ImageViews中的一个应从ChooserFragment过渡到DetailFragment。

在ChooserFragment布局中，我们需要唯一的transitionName属性：

```
<ImageView  
    android:id="@+id/image_first"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_first"  
    android:transitionName="fistImage" />  
  
<ImageView  
    android:id="@+id/image_second"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_second"  
    android:transitionName="secondImage" />
```

在ChooserFragments类中，我们需要传递被点击的View和一个ID给负责替换Fragments的父Activity（我们需要该ID来确定在DetailFragment中显示哪个图像资源）。如何向父Activity传递信息的详细内容肯定在其他文档中有介绍。

```
view.findViewById(R.id.image_first).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if (mCallback != null) {  
            mCallback.showDetailFragment(view, 1);  
        }  
    }  
});  
  
view.findViewById(R.id.image_second).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if (mCallback != null) {  
            mCallback.showDetailFragment(view, 2);  
        }  
    }  
});
```

在DetailFragment中，共享元素的ImageView也需要唯一的transitionName属性。

```
<ImageView  
    android:id="@+id/image_shared"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"
```

Chapter 193: Shared Element Transitions

Here you find examples for transition between Activities or Fragments using a shared element. An example for this behaviour is the Google Play Store App which translates an App's icon from the list to the App's details view.

Section 193.1: Shared Element Transition between two Fragments

In this example, one of two different ImageViews should be translated from the ChooserFragment to the DetailFragment.

In the ChooserFragment layout we need the unique transitionName attributes:

```
<ImageView  
    android:id="@+id/image_first"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_first"  
    android:transitionName="fistImage" />  
  
<ImageView  
    android:id="@+id/image_second"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_second"  
    android:transitionName="secondImage" />
```

In the ChooserFragments class, we need to pass the View which was clicked and an ID to the parent Activity which is handling the replacement of the fragments (we need the ID to know which image resource to show in the DetailFragment). How to pass information to a parent activity in detail is surely covered in another documentation.

```
view.findViewById(R.id.image_first).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if (mCallback != null) {  
            mCallback.showDetailFragment(view, 1);  
        }  
    }  
});  
  
view.findViewById(R.id.image_second).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if (mCallback != null) {  
            mCallback.showDetailFragment(view, 2);  
        }  
    }  
});
```

In the DetailFragment, the ImageView of the shared element also needs the unique transitionName attribute.

```
<ImageView  
    android:id="@+id/image_shared"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"
```

```
        android:transitionName="sharedImage" />
```

在DetailFragment的onCreateView()方法中，我们必须决定显示哪个图片资源
(如果不这样做，过渡后共享元素将消失)。

```
public static DetailFragment newInstance(Bundle args) {
    DetailFragment fragment = new DetailFragment();
    fragment.setArguments(args);
    return fragment;
}

@NoArgsConstructor
@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(R.layout.fragment_detail, container, false);

        ImageView sharedImage = (ImageView) view.findViewById(R.id.image_shared);

        // 检查应该显示哪个资源。
        int type = getArguments().getInt("type");

        // 根据类型显示图片。
        switch (type) {
            case 1:
                sharedImage.setBackgroundResource(R.drawable.ic_first);
                break;

            案例 2:
                sharedImage.setBackgroundResource(R.drawable.ic_second);
                break;
        }

        返回 view;
    }
}
```

父级 Activity 正在接回调并处理碎片的替换。

```
@Override
public void showDetailFragment(View sharedElement, int type) {
    // 获取当前显示的选择器碎片。
    Fragment chooserFragment = getFragmentManager().findFragmentById(R.id.fragment_container);

    // 设置 DetailFragment 并将类型作为参数传入。
    Bundle args = new Bundle();
    args.putInt("type", type);
    Fragment fragment = DetailFragment.newInstance(args);

    // 设置事务。
    FragmentTransaction transaction = getFragmentManager().beginTransaction();

    // 定义共享元素过渡。
    fragment.setSharedElementEnterTransition(new DetailsTransition());
    fragment.setSharedElementReturnTransition(new DetailsTransition());

    // 其余视图只是淡入/淡出效果。
    fragment.setEnterTransition(new Fade());
    chooserFragment.setExitTransition(new Fade());

    // 现在使用图片的视图和目标 transitionName 来定义共享元素。
    transaction.addSharedElement(sharedElement, "sharedImage");
}
```

```
        android:transitionName="sharedImage" />
```

In the onCreateView() method of the DetailFragment, we have to decide which image resource should be shown (if we don't do that, the shared element will disappear after the transition).

```
public static DetailFragment newInstance(Bundle args) {
    DetailFragment fragment = new DetailFragment();
    fragment.setArguments(args);
    return fragment;
}

@NoArgsConstructor
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(R.layout.fragment_detail, container, false);

    ImageView sharedImage = (ImageView) view.findViewById(R.id.image_shared);

    // Check which resource should be shown.
    int type = getArguments().getInt("type");

    // Show image based on the type.
    switch (type) {
        case 1:
            sharedImage.setBackgroundResource(R.drawable.ic_first);
            break;

        case 2:
            sharedImage.setBackgroundResource(R.drawable.ic_second);
            break;
    }

    return view;
}
}
```

The parent Activity is receiving the callbacks and handles the replacement of the fragments.

```
@Override
public void showDetailFragment(View sharedElement, int type) {
    // Get the chooser fragment, which is shown in the moment.
    Fragment chooserFragment = getFragmentManager().findFragmentById(R.id.fragment_container);

    // Set up the DetailFragment and put the type as argument.
    Bundle args = new Bundle();
    args.putInt("type", type);
    Fragment fragment = DetailFragment.newInstance(args);

    // Set up the transaction.
    FragmentTransaction transaction = getFragmentManager().beginTransaction();

    // Define the shared element transition.
    fragment.setSharedElementEnterTransition(new DetailsTransition());
    fragment.setSharedElementReturnTransition(new DetailsTransition());

    // The rest of the views are just fading in/out.
    fragment.setEnterTransition(new Fade());
    chooserFragment.setExitTransition(new Fade());

    // Now use the image's view and the target transitionName to define the shared element.
    transaction.addSharedElement(sharedElement, "sharedImage");
}
```

```
// 替换 fragment.  
transaction.replace(R.id.fragment_container, fragment, fragment.getClass().getSimpleName());  
  
// 启用带共享元素过渡的返回导航。  
transaction.addToBackStack(fragment.getClass().getSimpleName());  
  
// 最后按播放。  
transaction.commit();  
}
```

别忘了Transition本身。这个示例移动并缩放共享元素。

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)  
public class DetailsTransition extends TransitionSet {  
  
    public DetailsTransition() {  
        setOrdering(ORDERING_TOGETHER);  
        addTransition(new ChangeBounds()).  
            addTransition(new ChangeTransform()).  
            addTransition(new ChangeImageTransform());  
    }  
}
```

```
// Replace the fragment.  
transaction.replace(R.id.fragment_container, fragment, fragment.getClass().getSimpleName());  
  
// Enable back navigation with shared element transitions.  
transaction.addToBackStack(fragment.getClass().getSimpleName());  
  
// Finally press play.  
transaction.commit();  
}
```

Not to forget - the Transition itself. This example moves and scales the shared element.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)  
public class DetailsTransition extends TransitionSet {  
  
    public DetailsTransition() {  
        setOrdering(ORDERING_TOGETHER);  
        addTransition(new ChangeBounds()).  
            addTransition(new ChangeTransform()).  
            addTransition(new ChangeImageTransform());  
    }  
}
```

第194章：Android Things

第194.1节：控制伺服电机

本示例假设您的伺服电机具有以下特性，这些特性是典型的：

- 运动范围在0到180度之间
- 脉冲周期为20毫秒
- 最小脉冲长度为0.5毫秒
- 最大脉冲长度为2.5毫秒

您需要检查这些数值是否与您的硬件匹配，因为强制其超出指定的工作范围可能会损坏伺服电机。损坏的伺服电机反过来可能会损坏您的Android Things设备。示例中的ServoController类包含两个方法，setup()和setPosition()：

```
public class ServoController {  
    private double periodMs, maxTimeMs, minTimeMs;  
    private Pwm pin;  
  
    public void setup(String pinName) throws IOException {  
        periodMs = 20;  
        maxTimeMs = 2.5;  
        minTimeMs = 0.5;  
  
        PeripheralManagerService service = new PeripheralManagerService();  
        pin = service.openPwm(pinName);  
  
        pin.setPwmFrequencyHz(1000.0d / periodMs);  
        setPosition(90);  
        pin.setEnabled(true);  
    }  
  
    public void setPosition(double degrees) {  
        double pulseLengthMs = (degrees / 180.0 * (maxTimeMs - minTimeMs)) + minTimeMs;  
  
        if (pulseLengthMs < minTimeMs) {  
            pulseLengthMs = minTimeMs;  
        } else if (pulseLengthMs > maxTimeMs) {  
            pulseLengthMs = maxTimeMs;  
        }  
  
        double dutyCycle = pulseLengthMs / periodMs * 100.0;  
  
        Log.i(TAG, "占空比 = " + dutyCycle + " 脉冲长度 = " + pulseLengthMs);  
  
        try {  
            pin.setPwmDutyCycle(dutyCycle);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

您可以通过以下方式发现设备上支持PWM的引脚名称：

```
PeripheralManagerService service = new PeripheralManagerService();  
  
for (String pinName : service.getPwmList()) {
```

Chapter 194: Android Things

Section 194.1: Controlling a Servo Motor

This example assumes you have a servo with the following characteristics, which happen to be typical:

- movement between 0 and 180 degrees
- pulse period of 20 ms
- minimum pulse length of 0.5 ms
- maximum pulse length of 2.5 ms

You need to check if those values match your hardware, since forcing it to go outside its specified operating range can damage the servo. A damaged servo in turn has the potential to damage your Android Things device. The example ServoController class consists of two methods, setup() and setPosition():

```
public class ServoController {  
    private double periodMs, maxTimeMs, minTimeMs;  
    private Pwm pin;  
  
    public void setup(String pinName) throws IOException {  
        periodMs = 20;  
        maxTimeMs = 2.5;  
        minTimeMs = 0.5;  
  
        PeripheralManagerService service = new PeripheralManagerService();  
        pin = service.openPwm(pinName);  
  
        pin.setPwmFrequencyHz(1000.0d / periodMs);  
        setPosition(90);  
        pin.setEnabled(true);  
    }  
  
    public void setPosition(double degrees) {  
        double pulseLengthMs = (degrees / 180.0 * (maxTimeMs - minTimeMs)) + minTimeMs;  
  
        if (pulseLengthMs < minTimeMs) {  
            pulseLengthMs = minTimeMs;  
        } else if (pulseLengthMs > maxTimeMs) {  
            pulseLengthMs = maxTimeMs;  
        }  
  
        double dutyCycle = pulseLengthMs / periodMs * 100.0;  
  
        Log.i(TAG, "Duty cycle = " + dutyCycle + " pulse length = " + pulseLengthMs);  
  
        try {  
            pin.setPwmDutyCycle(dutyCycle);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

You can discover pin names that support PWM on your device as follows:

```
PeripheralManagerService service = new PeripheralManagerService();  
  
for (String pinName : service.getPwmList()) {
```

```
Log.i("ServoControlled", "找到PWM引脚: " + pinName);  
}
```

为了让你的舵机在80度和100度之间永远摆动，你可以简单地使用以下代码：

```
final ServoController servoController = new ServoController(pinName);  
  
Thread th = new Thread(new Runnable() {  
    @Override  
    public void run() {  
        while (true) {  
            try {  
                servoController.setPosition(80);  
                Thread.sleep(500);  
                servoController.setPosition(100);  
                Thread.sleep(500);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
});  
th.start();
```

你可以编译并部署上述所有代码，而无需实际将任何舵机连接到计算设备。关于接线，请参考你的计算设备引脚图（例如，树莓派3的引脚图可在这里找到）。

然后你需要将舵机连接到Vcc、Gnd和信号线。

```
Log.i("ServoControlled", "Pwm pin found: " + pinName);  
}
```

In order to make your servo swinging forever between 80 degrees and 100 degrees, you can simply use the following code:

```
final ServoController servoController = new ServoController(pinName);  
  
Thread th = new Thread(new Runnable() {  
    @Override  
    public void run() {  
        while (true) {  
            try {  
                servoController.setPosition(80);  
                Thread.sleep(500);  
                servoController.setPosition(100);  
                Thread.sleep(500);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
});  
th.start();
```

You can compile and deploy all of the above code without actually hooking any servo motors to the computing device. For the wiring, refer to your computing device pinout chart (e.g. a Raspberry Pi 3 pinout chart is available [here](#)).

Then you need to hook your servo to Vcc, Gnd, and signal.

第195章：库 Dagger 2： 应用中的依赖注入

Dagger 2, [如GitHub上所述](#), 是一种编译时演进的依赖注入方法。继承了Dagger 1.x中开始的方法, Dagger 2.x消除了所有反射, 并通过去除传统的ObjectGraph/Injector, 改用用户指定的@Component接口, 提升了代码的清晰度。

第195.1节：为对象创建@Module类和@Singleton注解

```
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class VehicleModule {

    @Provides @Singleton
    Motor provideMotor(){
        return new Motor();
    }

    @Provides @Singleton
    Vehicle provideVehicle(){
        return new Vehicle(new Motor());
    }
}
```

每个提供者（或方法）必须带有@Provides注解，且类必须带有@Module注解。
@Singleton注解表示该对象将只有一个实例。

第195.2节：依赖对象中的请求依赖

既然你已经有了不同模型的提供者，就需要请求它们。正如Vehicle需要Motor一样，
你必须在Vehicle的构造函数中添加@Inject注解，具体如下：

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

你可以使用@Inject注解在构造函数、字段或方法中请求依赖。在这个例子中，我将注入保持在构造函数中。

第195.3节：使用@Inject连接@Module

依赖提供者@Module与通过@Inject请求依赖的类之间的连接，是通过@Component实现的，
它是一个接口：

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
```

Chapter 195: Library Dagger 2: Dependency Injection in Applications

Dagger 2, [as explained on GitHub](#), is a compile-time evolution approach to dependency injection. Taking the approach started in Dagger 1.x to its ultimate conclusion, Dagger 2.x eliminates all reflection, and improves code clarity by removing the traditional ObjectGraph/Injector in favor of user-specified `@Component` interfaces.

Section 195.1: Create @Module Class and @Singleton annotation for Object

```
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class VehicleModule {

    @Provides @Singleton
    Motor provideMotor(){
        return new Motor();
    }

    @Provides @Singleton
    Vehicle provideVehicle(){
        return new Vehicle(new Motor());
    }
}
```

Every provider (or method) must have the `@Provides` annotation and the class must have the `@Module` annotation. The `@Singleton` annotation indicates that there will be only one instance of the object.

Section 195.2: Request Dependencies in Dependent Objects

Now that you have the providers for your different models, you need to request them. Just as Vehicle needs Motor, you have to add the `@Inject` annotation in the Vehicle constructor as follows:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

You can use the `@Inject` annotation to request dependencies in the constructor, fields, or methods. In this example, I'm keeping the injection in the constructor.

Section 195.3: Connecting @Modules with @Inject

The connection between the provider of dependencies, `@Module`, and the classes requesting them through `@Inject` is made using `@Component`, which is an interface:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
```

```
公共接口 VehicleComponent {  
    Vehicle 提供车辆();  
}
```

对于@Component注解，您必须指定将使用哪些模块。在此示例中使用了VehicleModule，该模块在此示例中定义。如果需要使用更多模块，只需用逗号分隔添加它们。

第195.4节：使用@Component接口获取对象

现在所有连接都已准备好，您必须获取该接口的实例并调用其方法以获取所需的对象：

```
VehicleComponent 组件 = Dagger_VehicleComponent.构建器().vehicleModule(新建  
VehicleModule()).构建();  
车辆 = 组件.提供车辆();  
Toast.makeText(this, String.valueOf(车辆.获取速度()), Toast.LENGTH_SHORT).显示();
```

当您尝试使用@Component注解创建接口的新对象时，必须使用前缀Dagger_<组件接口名称>来完成，在本例中为Dagger_VehicleComponent，然后使用构建器方法调用其中的每个模块。

```
public interface VehicleComponent {  
    Vehicle provideVehicle();  
}
```

For the @Component annotation, you have to specify which modules are going to be used. In this example VehicleModule is used, which is defined in this example. If you need to use more modules, then just add them using a comma as a separator.

Section 195.4: Using @Component Interface to Obtain Objects

Now that you have every connection ready, you have to obtain an instance of this interface and invoke its methods to obtain the object you need:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new  
VehicleModule()).build();  
vehicle = component.provideVehicle();  
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

When you try to create a new object of the interface with the @Component annotation, you have to do it using the prefix Dagger_<NameOfTheComponentInterface>, in this case Dagger_VehicleComponent, and then use the builder method to call every module within.

第196章：JCodec

第196.1节：入门

您可以通过maven自动获取JCodec。只需将以下代码片段添加到您的pom.xml中。

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

第196.2节：从视频中获取帧

从视频中获取单帧（仅支持MP4、ISO BMF、Quicktime容器中的AVC、H.264）：

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

从电影中获取一系列帧（仅支持MP4、ISO BMF、Quicktime容器中的AVC、H.264）：

```
double startSec = 51.632;
FileChannelWrapper ch = null;
try {
    ch = NIOUtils.readableFileChannel(new File("filename.mp4"));
    FrameGrab fg = new FrameGrab(ch);
    grab.seek(startSec);
    for (int i = 0; i < 100; i++) {
        ImageIO.write(grab.getFrame(), "png",
                     new File(System.getProperty("user.home"), String.format("Desktop/frame_%08d.png", i)));
    }
} finally {
    NIOUtils.closeQuietly(ch);
}
```

Chapter 196: JCodec

Section 196.1: Getting Started

You can get JCodec automatically with maven. For this just add below snippet to your pom.xml .

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Section 196.2: Getting frame from movie

Getting a single frame from a movie (supports only AVC, H.264 in MP4, ISO BMF, Quicktime container):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Getting a sequence of frames from a movie (supports only AVC, H.264 in MP4, ISO BMF, Quicktime container):

```
double startSec = 51.632;
FileChannelWrapper ch = null;
try {
    ch = NIOUtils.readableFileChannel(new File("filename.mp4"));
    FrameGrab fg = new FrameGrab(ch);
    grab.seek(startSec);
    for (int i = 0; i < 100; i++) {
        ImageIO.write(grab.getFrame(), "png",
                     new File(System.getProperty("user.home"), String.format("Desktop/frame_%08d.png", i)));
    }
} finally {
    NIOUtils.closeQuietly(ch);
}
```

第197章：使用模式格式化电话号码

本示例向您展示如何使用模式格式化电话号码

您需要在gradle中添加以下库。

```
compile 'com.googlecode.libphonenumber:libphonenumber:7.2.2'
```

第197.1节：模式 +1 (786) 1234 5678

给定一个规范化的电话号码，如 +178612345678，我们将使用提供的模式得到格式化的号码。

```
private String getFormattedNumber(String phoneNumber) {  
  
    PhoneNumberUtil phoneNumberUtil = PhoneNumberUtil.getInstance();  
  
    Phonemetadata.NumberFormat numberFormat = new Phonemetadata.NumberFormat();  
  
    numberFormat.pattern = "(\\d{3})(\\d{3})(\\d{4})";  
  
    numberFormat.format = "($1) $2-$3";  
  
    List<Phonemetadata.NumberFormat> newNumberFormats = new ArrayList<>();  
  
    newNumberFormats.add(numberFormat);  
  
    Phonenumbers.PhoneNumber phoneNumberPN = null;  
  
    try {  
        phoneNumberPN = phoneNumberUtil.parse(phoneNumber, Locale.US.getCountry());  
        phoneNumber = phoneNumberUtil.formatByPattern(phoneNumberPN,  
        PhoneNumberUtil.PhoneNumberFormat.INTERNATIONAL, newNumberFormats);  
  
    } catch (NumberParseException e) {  
        e.printStackTrace();  
    }  
  
    return phoneNumber;  
}
```

Chapter 197: Formatting phone numbers with pattern.

This example show you how to format phone numbers with a pattern

You will need the following library in your gradle.

```
compile 'com.googlecode.libphonenumber:libphonenumber:7.2.2'
```

Section 197.1: Patterns +1 (786) 1234 5678

Given a normalized phone number like +178612345678 we will get a formatted number with the provided pattern.

```
private String getFormattedNumber(String phoneNumber) {  
  
    PhoneNumberUtil phoneNumberUtil = PhoneNumberUtil.getInstance();  
  
    Phonemetadata.NumberFormat numberFormat = new Phonemetadata.NumberFormat();  
  
    numberFormat.pattern = "(\\d{3})(\\d{3})(\\d{4})";  
  
    numberFormat.format = "($1) $2-$3";  
  
    List<Phonemetadata.NumberFormat> newNumberFormats = new ArrayList<>();  
  
    newNumberFormats.add(numberFormat);  
  
    Phonenumbers.PhoneNumber phoneNumberPN = null;  
  
    try {  
        phoneNumberPN = phoneNumberUtil.parse(phoneNumber, Locale.US.getCountry());  
        phoneNumber = phoneNumberUtil.formatByPattern(phoneNumberPN,  
        PhoneNumberUtil.PhoneNumberFormat.INTERNATIONAL, newNumberFormats);  
  
    } catch (NumberParseException e) {  
        e.printStackTrace();  
    }  
  
    return phoneNumber;  
}
```

第198章：画笔

画笔是绘图所需的四个对象之一，另外三个是画布（Canvas，保存绘图调用）、位图（Bitmap，保存像素）和绘图基元（矩形、路径、位图等）

第198.1节：创建画笔

你可以使用以下三种构造函数之一创建新的画笔：

- `new Paint()` 使用默认设置创建
- `new Paint(int flags)` 使用标志创建
- `new Paint(Paint from)` 从另一个画笔复制设置

通常建议不要在onDraw()中创建画笔对象或任何其他对象，因为这可能导致性能问题。（Android Studio可能会警告你）相反，应将其设为全局变量，并在类构造函数中初始化，如下所示：

```
public class CustomView extends View {  
  
    private Paint paint;  
  
    public CustomView(Context context) {  
        super(context);  
        paint = new Paint();  
        //...  
    }  
  
    @Override  
    protected void onDraw(Canvas canvas) {  
        super.onDraw(canvas);  
        paint.setColor(0xFF000000);  
        // ...  
    }  
}
```

第198.2节：设置文本的Paint

文本绘制设置

- `setTypeface(Typeface typeface)` 设置字体样式。参见[Typeface](#)
- `setTextSize(int size)` 设置字体大小，单位为像素。
- `setColor(int color)` 设置绘制颜色，包括文本颜色。你也可以使用`setARGB(int a, int r, int g, int b)`和`setAlpha(int alpha)`
- `setLetterSpacing(float size)` 设置字符间距，单位为ems。默认值为0，负值会使文本收紧，正值会使文本扩展。
- `setTextAlign(Paint.Align align)` 设置文本相对于其原点的对齐方式。`Paint.Align.LEFT`会将文本绘制在原点的右侧，`RIGHT`会绘制在左侧，`CENTER`会使文本居中于原点（水平）
- `setTextSkewX(float skewX)` 这可以视为伪斜体。SkewX表示文本底部的水平偏移。（斜体使用-0.25）
- `setStyle(Paint.Style style)` 填充文本 `FILL`，描边文本 `STROKE`，或两者兼有 `FILL_AND_STROKE`

请注意，您可以使用`TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())`将SP或DP转换为像素。

Chapter 198: Paint

A paint is one of the four objects needed to draw, along with a Canvas (holds drawing calls), a Bitmap (holds the pixels), and a drawing primitive (Rect, Path, Bitmap...)

Section 198.1: Creating a Paint

You can create a new paint with one of these 3 constructors:

- `new Paint()` Create with default settings
- `new Paint(int flags)` Create with flags
- `new Paint(Paint from)` Copy settings from another paint

It is generally suggested to never create a paint object, or any other object in onDraw() as it can lead to performance issues. (Android Studio will probably warn you) Instead, make it global and initialize it in your class constructor like so:

```
public class CustomView extends View {  
  
    private Paint paint;  
  
    public CustomView(Context context) {  
        super(context);  
        paint = new Paint();  
        //...  
    }  
  
    @Override  
    protected void onDraw(Canvas canvas) {  
        super.onDraw(canvas);  
        paint.setColor(0xFF000000);  
        // ...  
    }  
}
```

Section 198.2: Setting up Paint for text

Text drawing settings

- `setTypeface(Typeface typeface)` Set the font face. See [Typeface](#)
- `setTextSize(int size)` Set the font size, in pixels.
- `setColor(int color)` Set the paint drawing color, including the text color. You can also use `setARGB(int a, int r, int g, int b)` and `setAlpha(int alpha)`
- `setLetterSpacing(float size)` Set the spacing between characters, in ems. Default value is 0, a negative value will tighten the text, while a positive one will expand it.
- `setTextAlign(Paint.Align align)` Set text alignment relative to its origin. `Paint.Align.LEFT` will draw it to the right of the origin, `RIGHT` will draw it to the left, and `CENTER` will draw it centered on the origin (horizontally)
- `setTextSkewX(float skewX)` This could be considered as fake italic. SkewX represents the horizontal offset of the text bottom. (use -0.25 for italic)
- `setStyle(Paint.Style style)` Fill text `FILL`, Stroke text `STROKE`, or both `FILL_AND_STROKE`

Note that you can use `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` to convert from SP or DP to pixels.

测量文本

- `float width = paint.measureText(String text)` 测量文本宽度
- `float height = paint.ascent()` 测量文本高度
- `paint.getTextBounds(String text, int start, int end, Rect bounds)` 存储文本尺寸。您需要分配Rect，不能为null：

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

还有其他测量方法，但这三种方法应满足大多数需求。

第198.3节：设置Paint以绘制形状

- `setStyle(Paint.Style style)` 填充形状 FILL，描边形状 STROKE，或两者 FILL_AND_STROKE
- `setColor(int color)` 设置绘图颜色。您也可以使用`setARGB(int a, int r, int g, int b)` 和`setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` 设置线帽，选项为ROUND、SQUARE或BUTT（无）。详见[this](#)。
- `setStrokeJoin(Paint.Join join)` 设置线连接方式，选项为MITER（尖角）、ROUND或BEVEL。详见[this](#)。
- `setStrokeMiter(float miter)` 设置斜接连接限制。此设置可防止斜接连接无限延伸，超过x像素后转为斜面连接。详见[this](#)。
- `setStrokeWidth(float width)` 设置描边宽度。设置为0时，将以发丝线模式绘制，与画布矩阵无关。（始终为1像素）

第198.4节：设置标志

您可以在构造函数中设置以下标志，或使用`setFlags(int flags)`进行设置

- `Paint.ANTI_ALIAS_FLAG` 启用抗锯齿，平滑绘制效果。
- `Paint.DITHER_FLAG` 启用抖动。如果颜色精度高于设备，这将发生。
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` 启用位图字体的使用。
- `Paint.FAKE_BOLD_TEXT_FLAG` 将以伪粗体效果绘制文本，可替代使用粗体字体。一些字体有样式化粗体，伪粗体则没有
- `Paint.FILTER_BITMAP_FLAG` 影响位图变换时的采样。
- `Paint.HINTING_OFF, Paint.HINTING_ON` 切换字体提示，详见[this](#)
- `Paint.LINEAR_TEXT_FLAG` 禁用字体缩放，绘制操作将被缩放
- `Paint.SUBPIXEL_TEXT_FLAG` 文本将使用子像素精度计算。
- `Paint.STRIKE_THRU_TEXT_FLAG` 绘制的文本将带有删除线
- `Paint.UNDERLINE_TEXT_FLAG` 绘制的文本将带有下划线

你可以这样添加标志和移除标志：

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG); // 添加标志
paint.setFlags(paint.getFlags() & ~Paint.FLAG); // 移除标志
```

尝试移除不存在的标志或添加已存在的标志不会有任何变化。另请注意，大多数标志也可以通过`set<Flag>(boolean enabled)`来设置，例如`setAntialias(true)`。

你可以使用`paint.reset()`将画笔重置为默认设置。唯一的默认标志是`EMBEDDED_BITMAP_TEXT_FLAG`。即使你使用`new Paint(0)`，它也会被设置，你将拥有

Measuring text

- `float width = paint.measureText(String text)` Measure the width of text
- `float height = paint.ascent()` Measure the height of text
- `paint.getTextBounds(String text, int start, int end, Rect bounds)` Stores the text dimensions. You have allocate the Rect, it cannot be null:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

There are other methods for measuring, however these three should fit most purposes.

Section 198.3: Setting up Paint for drawing shapes

- `setStyle(Paint.Style style)` Filled shape FILL, Stroke shape STROKE, or both FILL_AND_STROKE
- `setColor(int color)` Set the paint drawing color. You can also use `setARGB(int a, int r, int g, int b)` and `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` Set line caps, either ROUND, SQUARE, or BUTT (none) See [this](#).
- `setStrokeJoin(Paint.Join join)` Set line joins, either MITER (pointy), ROUND, or BEVEL. See [this](#).
- `setStrokeMiter(float miter)` Set miter join limit. This can prevent miter join from going on indefinitely, turning it into a bevel join after x pixels. See [this](#).
- `setStrokeWidth(float width)` Set stroke width. 0 will draw in hairline mode, independant of the canvas matrix. (always 1 pixel)

Section 198.4: Setting flags

You can set the following flags in the constructor, or with `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Enable antialiasing, smooths the drawing.
- `Paint.DITHER_FLAG` Enable dithering. If color precision is higher than the device's, [this will happen](#).
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Enables the use of bitmap fonts.
- `Paint.FAKE_BOLD_TEXT_FLAG` will draw text with a fake bold effect, can be used instead of using a bold typeface. Some fonts have styled bold, [fake bold won't](#)
- `Paint.FILTER_BITMAP_FLAG` Affects the sampling of bitmaps when transformed.
- `Paint.HINTING_OFF, Paint.HINTING_ON` Toggles font hinting, see [this](#)
- `Paint.LINEAR_TEXT_FLAG` Disables font scaling, draw operations are scaled instead
- `Paint.SUBPIXEL_TEXT_FLAG` Text will be computed using subpixel accuracy.
- `Paint.STRIKE_THRU_TEXT_FLAG` Text drawn will be striked
- `Paint.UNDERLINE_TEXT_FLAG` Text drawn will be underlined

You can add a flag and remove flags like this:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG); // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG); // Remove flag
```

Trying to remove a flag that isn't there or adding a flag that is already there won't change anything. Also note that most flags can also be set using `set<Flag>(boolean enabled)`, for example `setAntialias(true)`.

You can use `paint.reset()` to reset the paint to its default settings. The only default flag is `EMBEDDED_BITMAP_TEXT_FLAG`. It will be set even if you use `new Paint(0)`, you will have

第199章：什么是ProGuard？它在Android中的用途是什么？

Proguard是免费的Java类文件压缩器、优化器、混淆器和预验证器。它检测并移除未使用的类、字段、方法和属性。它优化字节码并移除未使用的指令。它使用简短且无意义的名称重命名剩余的类、字段和方法。

第199.1节：使用ProGuard缩减代码和资源

为了使您的APK文件尽可能小，您应该启用缩减功能，以删除发布构建中未使用的代码和资源。本页介绍了如何操作以及如何指定在构建过程中保留或丢弃哪些代码和资源。

代码缩减功能由ProGuard提供，它可以检测并移除打包应用中未使用的类、字段、方法和属性，包括来自包含的代码库的内容（这使其成为解决64k引用限制的有力工具）。ProGuard还会优化字节码，删除未使用的代码指令，并用简短的名称混淆剩余的类、字段和方法。混淆后的代码使您的APK难以被逆向工程，这在应用使用安全敏感功能（如许可验证）时尤其重要。

资源缩减功能由Android Gradle插件提供，它会移除打包应用中未使用的资源，包括代码库中未使用的资源。它与代码缩减协同工作，一旦未使用的代码被移除，任何不再被引用的资源也可以安全地删除。

缩减您的代码

要启用ProGuard代码缩减，请在您的build.gradle文件中相应的构建类型中添加minifyEnabled true。

请注意，代码缩减会降低构建速度，因此如果可能，您应避免在调试构建中使用它。然而，重要的是您在用于测试的最终APK中启用代码缩减，因为如果您未充分自定义保留哪些代码，可能会引入错误。

例如，以下build.gradle文件片段启用了发布构建的代码缩减：

```
android {  
    buildTypes {  
        release {  
            minifyEnabled true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
}
```

除了minifyEnabled属性外，proguardFiles属性定义了ProGuard规则：

getDefaultProguardFile('proguard-android.txt') 方法从 AndroidSDK tools/proguard/文件夹获取默认的 ProGuard 设置。提示：为了进一步缩减代码，可以尝试位于同一位置的proguard-android-optimize.txt文件。它包含相同的 ProGuard 规则，但带有其他优化，这些优化在字节码级别—方法内部及跨方法—进行分析，以进一步减小 APK 大小并帮助其更快运行。proguard-rules.pro文件是您可以添加自定义 ProGuard 规则的地方。默认情况下，该文件位于模块根目录（与 build.gradle 文件同级）。

Chapter 199: What is ProGuard? What is use in Android?

Proguard is free Java class file shrinker, optimizer, obfuscator, and preverifier. It detects and removes unused classes, fields, methods, and attributes. It optimizes bytecode and removes unused instructions. It renames the remaining classes, fields, and methods using short meaningless names.

Section 199.1: Shrink your code and resources with proguard

To make your APK file as small as possible, you should enable shrinking to remove unused code and resources in your release build. This page describes how to do that and how to specify what code and resources to keep or discard during the build.

Code shrinking is available with ProGuard, which detects and removes unused classes, fields, methods, and attributes from your packaged app, including those from included code libraries (making it a valuable tool for working around the 64k reference limit). ProGuard also optimizes the bytecode, removes unused code instructions, and obfuscates the remaining classes, fields, and methods with short names. The obfuscated code makes your APK difficult to reverse engineer, which is especially valuable when your app uses security-sensitive features, such as licensing verification.

Resource shrinking is available with the Android plugin for Gradle, which removes unused resources from your packaged app, including unused resources in code libraries. It works in conjunction with code shrinking such that once unused code has been removed, any resources no longer referenced can be safely removed as well.

Shrink Your Code

To enable code shrinking with ProGuard, add minifyEnabled true to the appropriate build type in your build.gradle file.

Be aware that code shrinking slows down the build time, so you should avoid using it on your debug build if possible. However, it's important that you do enable code shrinking on your final APK used for testing, because it might introduce bugs if you do not sufficiently customize which code to keep.

For example, the following snippet from a build.gradle file enables code shrinking for the release build:

```
android {  
    buildTypes {  
        release {  
            minifyEnabled true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
}
```

In addition to the minifyEnabled property, the proguardFiles property defines the ProGuard rules:

The getDefaultProguardFile('proguard-android.txt') method gets the default ProGuard settings from the Android SDK tools/proguard/ folder. Tip: For even more code shrinking, try the proguard-android-optimize.txt file that's in the same location. It includes the same ProGuard rules, but with other optimizations that perform analysis at the bytecode level—inside and across methods—to reduce your APK size further and help it run faster. The proguard-rules.pro file is where you can add custom ProGuard rules. By default, this file is located at the root of

要为每个构建变体添加特定的更多 ProGuard 规则，请在相应的productFlavor块中添加另一个 proguardFiles 属性。例如，以下 Gradle 文件为 flavor2 产品风味添加了 flavor2-rules.pro。现在 flavor2 使用所有三个 ProGuard 规则，因为 release 块中的规则也被应用。

```
android {  
    ...  
    buildTypes {  
        release {  
            minifyEnabled true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
    productFlavors {  
        flavor1 {  
        }  
        flavor2 {  
            proguardFile 'flavor2-rules.pro'  
        }  
    }  
}
```

the module (next to the build.gradle file). To add more ProGuard rules that are specific to each build variant, add another proguardFiles property in the corresponding productFlavor block. For example, the following Gradle file adds flavor2-rules.pro to the flavor2 product flavor. Now flavor2 uses all three ProGuard rules because those from the release block are also applied.

```
android {  
    ...  
    buildTypes {  
        release {  
            minifyEnabled true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
    productFlavors {  
        flavor1 {  
        }  
        flavor2 {  
            proguardFile 'flavor2-rules.pro'  
        }  
    }  
}
```

第200章：创建Android自定义ROM

第200.1节：准备构建环境！

在你开始构建任何东西之前，需要先准备好你的机器。为此，你需要安装大量的库和模块。最推荐的Linux发行版是Ubuntu，因此本例将重点介绍在Ubuntu上安装所需的一切。

安装Java

```
首先，添加以下个人软件包档案（PPA）：sudo apt-add-repository ppa:openjdk-r/ppa
```

```
然后，通过执行以下命令更新源：sudo apt-get update
```

安装额外依赖

所有必需的额外依赖可以通过以下命令安装：

```
sudo apt-get install git-core python gnupg flex bison gperf libSDL1.2-dev libesd0-dev libwxgtk2.8-dev squashfs-tools build-essential zip curl libncurses5-dev zlib1g-dev openjdk-8-jre openjdk-8-jdk pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool g++-multilib lib32z1-dev lib32ncurses5-dev gcc-multilib liblz4-* pngquant ncurses-dev texinfo gcc gperf patch libtool automake g++ gawk subversion expat libexpat1-dev python-all-dev binutils-static bc libcloog-isl-dev libcap-dev autoconf libgmp-dev build-essential gcc-multilib g++-multilib pkg-config libmpc-dev libmpfr-dev lzma* liblzma* w3m android-tools-adb maven ncftp figlet
```

准备开发环境

现在所有依赖项都已安装，让我们通过执行以下命令来准备开发系统：

```
sudo curl --create-dirs -L -o /etc/udev/rules.d/51-android.rules -O -L https://raw.githubusercontent.com/snowdream/51-android/master/51-android.rules
sudo chmod 644 /etc/udev/rules.d/51-android.rules
sudo chown root /etc/udev/rules.d/51-android.rules
sudo service udev restart
adb kill-server
sudo killall adb
```

最后，让我们通过以下命令设置缓存和仓库：

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

请注意：我们也可以通过运行由Akhil Narang (akhilnarang) 制作的自动化脚本来完成此设置，他是 Resurrection Remix OS 的维护者之一。这些脚本可以在GitHub上找到。

Chapter 200: Create Android Custom ROMs

Section 200.1: Making Your Machine Ready for Building!

Before you can build anything, you are required to make your machine ready for building. For this you need to install a lot of libraries and modules. The most recommended Linux distribution is Ubuntu, so this example will focus on installing everything that is needed on Ubuntu.

Installing Java

First, add the following Personal Package Archive (PPA): `sudo apt-add-repository ppa:openjdk-r/ppa`.

Then, update the sources by executing: `sudo apt-get update`.

Installing Additional Dependencies

All required additional dependencies can be installed by the following command:

```
sudo apt-get install git-core python gnupg flex bison gperf libSDL1.2-dev libesd0-dev libwxgtk2.8-dev squashfs-tools build-essential zip curl libncurses5-dev zlib1g-dev openjdk-8-jre openjdk-8-jdk pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool g++-multilib lib32z1-dev lib32ncurses5-dev gcc-multilib liblz4-* pngquant ncurses-dev texinfo gcc gperf patch libtool automake g++ gawk subversion expat libexpat1-dev python-all-dev binutils-static bc libcloog-isl-dev libcap-dev autoconf libgmp-dev build-essential gcc-multilib g++-multilib pkg-config libmpc-dev libmpfr-dev lzma* liblzma* w3m android-tools-adb maven ncftp figlet
```

Preparing the system for development

Now that all the dependencies are installed, let us prepare the system for development by executing:

```
sudo curl --create-dirs -L -o /etc/udev/rules.d/51-android.rules -O -L https://raw.githubusercontent.com/snowdream/51-android/master/51-android.rules
sudo chmod 644 /etc/udev/rules.d/51-android.rules
sudo chown root /etc/udev/rules.d/51-android.rules
sudo service udev restart
adb kill-server
sudo killall adb
```

Finally, let us set up the cache and the repo by the following commands:

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Please note: We can also achieve this setup by running the automated scripts made by Akhil Narang (akhilnarang), one of the maintainers of [Resurrection Remix OS](#). These scripts can be found [on GitHub](#).

第201章：Android的Genymotion

Genymotion是一款快速的第三方模拟器，可以替代默认的Android模拟器。在某些情况下，它的表现与实际设备开发一样好甚至更好！

第201.1节：安装Genymotion免费版

步骤1 - 安装VirtualBox

根据您的操作系统下载并安装VirtualBox。运行Genymotion需要它。

步骤2 - 下载Genymotion

访问Genymotion下载页面，根据您的操作系统下载Genymotion。

注意：您需要创建一个新账户或使用已有账户登录。

步骤3 - 安装Genymotion

如果在Linux上，请参考此答案，以安装和运行.bin文件。

步骤4 - 安装Genymotion的模拟器

- 运行Genymotion
- 点击顶部栏的添加按钮。
- 使用您的账户登录，您将能够浏览可用的模拟器。
- 选择并安装您需要的模拟器。

步骤5 - 将Genymotion与Android Studio

Genymotion集成，Genymotion可以通过插件与Android Studio集成，以下是在Android Studio中安装插件的步骤

- 进入文件/设置（Windows和Linux）或Android Studio/偏好设置（Mac OS X）
- 选择插件并点击浏览仓库。
- 右键点击Genymotion，选择下载并安装。

您现在应该能看到插件图标，参见此

注意，您可能需要通过点击视图 > 工具栏来显示工具栏。

步骤6 - 从Android Studio运行Genymotion

- 进入文件/设置（Windows和Linux）或Android Studio/偏好设置（Mac OS X）
- 进入其他设置/Genymotion，添加Genymotion文件夹的路径并应用更改。

现在你应该能够通过点击插件图标，选择已安装的模拟器，然后按下开始按钮来运行Genymotion的模拟器了！

Chapter 201: Genymotion for android

Genymotion is a fast third-party emulator that can be used instead of the default Android emulator. In some cases it's as good as or better than developing on actual devices!

Section 201.1: Installing Genymotion, the free version

Step 1 - installing VirtualBox

Download and install [VirtualBox](#) according to your operating system., it is required to run Genymotion.

Step 2 - downloading Genymotion

Go to the [Genymotion download page](#) and download Genymotion according to your operating system.

Note: you will need to create a new account OR log-in with your account.

Step 3 - Installing Genymotion

if on Linux then refer to this [answer](#), to install and run a .bin file.

Step 4 - Installing Genymotion's emulators

- run Genymotion
- Press on the Add button (in top bar).
- Log-In with your account and you will be able to browse the available emulators.
- select and Install what you need.

Step 5 - Integrating genymotion with Android Studio

Genymotion, can be integrated with Android Studio via a plugin, here the steps to install it in Android Studio

- go to File/Settings (for Windows and Linux) or to Android Studio/Preferences (for Mac OS X)
- Select Plugins and click Browse Repositories.
- Right-click on Genymotion and click Download and install.

You should now be able to see the plugin icon, see this [image](#)

Note, you might want to display the toolbar by clicking View > Toolbar.

Step 6 - Running Genymotion from Android Studio

- go to File/Settings (for Windows and Linux) or to Android Studio/Preferences (for Mac OS X)
- go to Other Settings/Genymotion and add the path of Genymotion's folder and apply your changes.

Now you should be able to run Genymotion's emulator by pressing the plugin icon and selecting an installed emulator and then press start button!

第201.2节：Genymotion上的谷歌框架

如果开发者想测试谷歌地图或任何其他谷歌服务，如Gmail、YouTube、Google Drive等，那么他们首先需要在Genymotion上安装谷歌框架。以下是步骤：

[4.4 Kitkat](#)

[5.0 Lollipop](#)

[5.1 Lollipop](#)

[6.0 Marshmallow](#)

[7.0 Nougat](#)

[7.1 Nougat \(WebView补丁\)](#)

1. 从上述链接下载
2. 只需将下载的压缩文件拖放到 Genymotion 中并重启
3. 添加谷歌账号，下载“Google Play 音乐”并运行。

参考：

[关于此主题的 Stack Overflow 问题](#)

Section 201.2: Google framework on Genymotion

If developers want to test Google Maps or any other Google service like Gmail, Youtube, Google drive etc. then they first need to install Google framework on Genymotion. Here are the steps:

[4.4 Kitkat](#)

[5.0 Lollipop](#)

[5.1 Lollipop](#)

[6.0 Marshmallow](#)

[7.0 Nougat](#)

[7.1 Nougat \(webview patch\)](#)

1. Download from above link
2. Just drag & drop downloaded zip file to genymotion and restart
3. Add google account and download "Google Play Music" and Run.

Reference:

[Stack overflow question on this topic](#)

第202章：ConstraintSet（约束集）

此类允许您以编程方式定义一组用于ConstraintLayout的约束。它让您创建并保存约束，并将其应用于现有的ConstraintLayout。

第202.1节：使用 ConstraintLayout 的 ConstraintSet 编程实现

```
import android.content.Context;
import android.os.Bundle;
import android.support.constraint.ConstraintLayout;
import android.support.constraint.ConstraintSet;
import android.support.transition.TransitionManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    ConstraintSet mConstraintSet1 = new ConstraintSet(); // 创建一个约束集
    ConstraintSet mConstraintSet2 = new ConstraintSet(); // 创建一个约束集
    ConstraintLayout mConstraintLayout; // 缓存 ConstraintLayout
    boolean mOld = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Context context = this;
        mConstraintSet2.clone(context, R.layout.state2); // 从布局获取约束
        setContentView(R.layout.state1);
        mConstraintLayout = (ConstraintLayout) findViewById(R.id.activity_main);
        mConstraintSet1.clone(mConstraintLayout); // 从ConstraintSet获取约束
    }

    public void foo(View view) {
        TransitionManager.beginDelayedTransition(mConstraintLayout);
        if (mOld != !mOld) {
            mConstraintSet1.applyTo(mConstraintLayout); // 设置新约束
        } else {
            mConstraintSet2.applyTo(mConstraintLayout); // 设置新约束
        }
    }
}
```

Chapter 202: ConstraintSet

This class allows you to define programmatically a set of constraints to be used with ConstraintLayout. It lets you create and save constraints, and apply them to an existing ConstraintLayout.

Section 202.1: ConstraintSet with ConstraintLayout Programmatically

```
import android.content.Context;
import android.os.Bundle;
import android.support.constraint.ConstraintLayout;
import android.support.constraint.ConstraintSet;
import android.support.transition.TransitionManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    ConstraintSet mConstraintSet1 = new ConstraintSet(); // create a Constraint Set
    ConstraintSet mConstraintSet2 = new ConstraintSet(); // create a Constraint Set
    ConstraintLayout mConstraintLayout; // cache the ConstraintLayout
    boolean mOld = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Context context = this;
        mConstraintSet2.clone(context, R.layout.state2); // get constraints from layout
        setContentView(R.layout.state1);
        mConstraintLayout = (ConstraintLayout) findViewById(R.id.activity_main);
        mConstraintSet1.clone(mConstraintLayout); // get constraints from ConstraintSet
    }

    public void foo(View view) {
        TransitionManager.beginDelayedTransition(mConstraintLayout);
        if (mOld != !mOld) {
            mConstraintSet1.applyTo(mConstraintLayout); // set new constraints
        } else {
            mConstraintSet2.applyTo(mConstraintLayout); // set new constraints
        }
    }
}
```

第203章：CleverTap

CleverTap 提供的分析和参与 SDK 快速技巧 - 安卓

第203.1节：设置调试级别

在您的自定义应用程序类中，重写 `onCreate()`方法，添加以下代码行：

```
CleverTapAPI.setDebugLevel(1);
```

第203.2节：获取SDK实例以记录事件

```
CleverTapAPI cleverTap;  
try {  
    cleverTap = CleverTapAPI.getInstance(getApplicationContext());  
} catch (CleverTapMetaDataNotFoundException e) {  
    // 如果您未在AndroidManifest.xml中指定CleverTap账户ID或令牌，则会抛出此异常  
} catch (CleverTapPermissionsNotSatisfied e) {  
    // 如果您未在AndroidManifest.xml中请求所需权限，则会抛出此异常  
}
```

Chapter 203: CleverTap

Quick hacks for the analytics and engagement SDK provided by CleverTap - Android

Section 203.1: Setting the debug level

In your custom application class, override the `onCreate()` method, add the line below:

```
CleverTapAPI.setDebugLevel(1);
```

Section 203.2: Get an instance of the SDK to record events

```
CleverTapAPI cleverTap;  
try {  
    cleverTap = CleverTapAPI.getInstance(getApplicationContext());  
} catch (CleverTapMetaDataNotFoundException e) {  
    // thrown if you haven't specified your CleverTap Account ID or Token in your AndroidManifest.xml  
} catch (CleverTapPermissionsNotSatisfied e) {  
    // thrown if you haven't requested the required permissions in your AndroidManifest.xml  
}
```

第204章：将库发布到Maven仓库

第204.1节：发布.aar文件到Maven

为了发布到Maven格式的仓库，可以使用Gradle的“maven-publish”插件。

该插件应添加到库模块的build.gradle文件中。

```
apply plugin: 'maven-publish'
```

你还需要在build.gradle文件中定义发布及其身份属性。这些身份属性将显示在生成的pom文件中，未来导入该发布时也会使用它们。你还需要定义想要发布的构件，例如我只想发布构建库后生成的.aar文件。

```
publishing {  
    publications {  
        myPulication(MavenPublication) {  
            groupId 'com.example.project'  
            version '1.0.2'  
            artifactId 'myProject'  
            artifact("$buildDir/outputs/aar/myProject.aar")  
        }  
    }  
}
```

你还需要定义你的仓库URL

```
publishing{  
    仓库 {  
        maven {  
            url "http://www.myrepository.com"  
        }  
    }  
}
```

这是完整的库build.gradle文件

```
应用插件: 'com.android.library'  
应用插件: 'maven-publish'  
  
构建脚本 {  
    ...  
}  
android {  
    ...  
}  
publishing {  
    publications {  
        myPulication(MavenPublication) {  
            groupId 'com.example.project'  
            version '1.0.2'  
            artifactId 'myProject'  
            artifact("$buildDir/outputs/aar/myProject.aar")  
        }  
    }  
}
```

Chapter 204: Publish a library to Maven Repositories

Section 204.1: Publish .aar file to Maven

In order to publish to a repository in Maven format ,“maven-publish” plugin for gradle can be used.

The plugin should be added to build.gradle file in library module.

```
apply plugin: 'maven-publish'
```

You should define the publication and its identity attributes in build.gradle file too. This identity attributes will be shown in the generated pom file and in future for importing this publication you will use them. You also need to define which artifacts you want to publish,for example i just want to publish generated .aar file after building the library.

```
publishing {  
    publications {  
        myPulication(MavenPublication) {  
            groupId 'com.example.project'  
            version '1.0.2'  
            artifactId 'myProject'  
            artifact("$buildDir/outputs/aar/myProject.aar")  
        }  
    }  
}
```

You will also need to define your repository url

```
publishing{  
    repositories {  
        maven {  
            url "http://www.myrepository.com"  
        }  
    }  
}
```

Here is full library build.gradle file

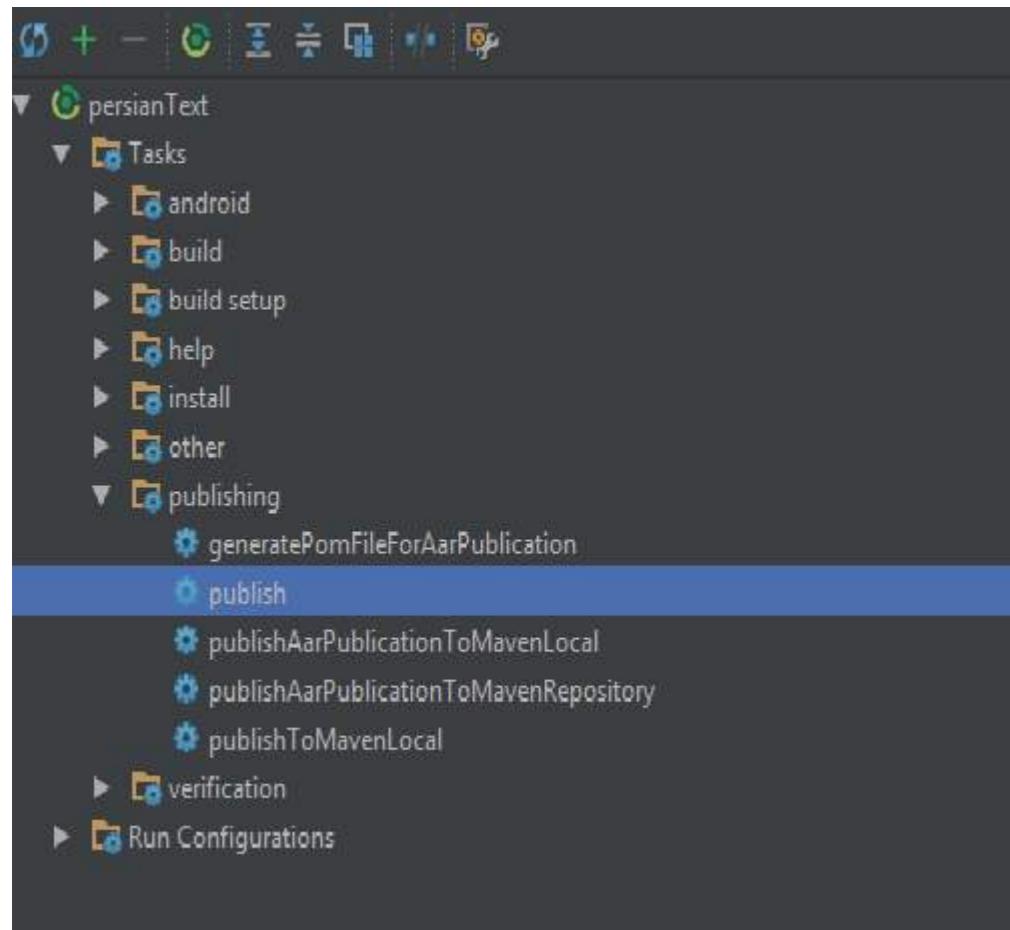
```
apply plugin: 'com.android.library'  
apply plugin: 'maven-publish'  
  
buildscript {  
    ...  
}  
android {  
    ...  
}  
publishing {  
    publications {  
        myPulication(MavenPublication) {  
            groupId 'com.example.project'  
            version '1.0.2'  
            artifactId 'myProject'  
            artifact("$buildDir/outputs/aar/myProject.aar")  
        }  
    }  
}
```

```
    }
仓库 {
maven {
url "http://www.myrepository.com"
}
}
```

发布时你可以运行gradle控制台命令

```
gradle publish
```

或者你可以从gradle任务面板运行

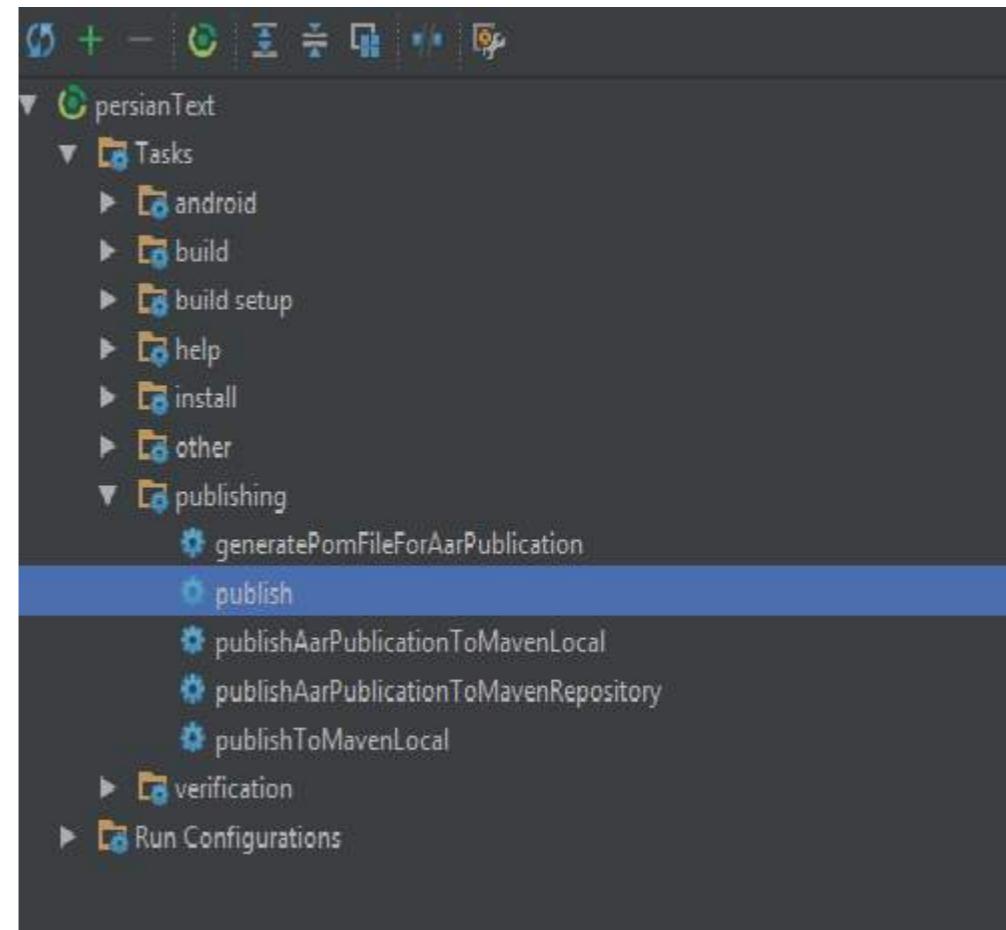


```
}
repositories {
    maven {
        url "http://www.myrepository.com"
    }
}
```

For publishing you can run gradle console command

```
gradle publish
```

or you can run from gradle tasks panel



第205章：adb shell

参数	详细信息
-e	选择转义字符，或“none”；默认“~”
-n	不从标准输入读取
-T	禁用PTY分配
-t	强制PTY分配
-x	禁用远程退出代码和标准输出/标准错误分离

adb shell 在目标设备或模拟器中打开一个 Linux shell。这是通过adb控制Android设备最强大且多功能的方式。

由于示例数量达到上限，其中许多涉及adb shell命令，本主题从ADB（Android调试桥）中拆分出来。

第205.1节：授予和撤销API 23及以上权限

一行命令，帮助授予或撤销易受攻击的权限。

- 授予

```
adb shell pm grant <sample.package.id> android.permission.<PERMISSION_NAME>
```

- 撤销

```
adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
```

- 安装时一次性授予所有运行时权限 (-g)

```
adb install -g /path/to/sample_package.apk
```

第205.2节：通过ADB向Android设备发送文本、按键和触摸事件

执行以下命令将文本插入到具有焦点的视图中（如果支持文本输入）

版本 ≥ 6.0

在SDK 23及以上版本发送文本

```
adb shell "input keyboard text 'Paste text on Android Device'"
```

如果已经通过adb连接到您的设备：

```
input text 'Paste text on Android Device'
```

版本 < 6.0

在SDK 23之前发送文本

```
adb shell "input keyboard text 'Paste%stext%son%sAndroid%sDevice'"
```

Chapter 205: adb shell

Parameter	Details
-e	choose escape character, or "none"; default '~'
-n	don't read from stdin
-T	disable PTY allocation
-t	force PTY allocation
-x	disable remote exit codes and stdout/stderr separation

adb shell opens a Linux shell in a target device or emulator. It is the most powerful and versatile way to control an Android device via adb.

This topic was split from ADB (Android Debug Bridge) due to reaching the limit of examples, many of which were involving adb shell command.

Section 205.1: Granting & revoking API 23+ permissions

A one-liner that helps granting or revoking vulnerable permissions.

- granting

```
adb shell pm grant <sample.package.id> android.permission.<PERMISSION_NAME>
```

- revoking

```
adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
```

- Granting all run-time permissions at a time on installation (-g)

```
adb install -g /path/to/sample_package.apk
```

Section 205.2: Send text, key pressed and touch events to Android Device via ADB

execute the following command to insert the text into a view with a focus (if it supports text input)

Version ≥ 6.0

Send text on SDK 23+

```
adb shell "input keyboard text 'Paste text on Android Device'"
```

If already connected to your device via adb:

```
input text 'Paste text on Android Device'
```

Version < 6.0

Send text prior to SDK 23

```
adb shell "input keyboard text 'Paste%stext%son%sAndroid%sDevice'"
```

输入中不接受空格, 请用 %s 替换。

发送事件

模拟按下硬件电源键

```
adb shell input keyevent 26
```

或者

```
adb shell input keyevent POWER
```

即使没有硬件按键, 您仍然可以使用keyevent执行等效操作

```
adb shell input keyevent CAMERA
```

发送触摸事件作为输入

```
adb shell input tap X点 Y点
```

发送滑动事件作为输入

```
adb shell input swipe X点1 Y点1 X点2 Y点2 [持续时间*]
```

*持续时间为可选, 默认=300毫秒。 来源 [_____](#)

通过在开发者选项中启用指针位置获取X和Y坐标。

ADB示例shell脚本

在Ubuntu中运行脚本, 创建script.sh, 右键文件并添加读写权限, 勾选**允许将文件作为程序执行**。

打开终端模拟器并运行命令 ./script.sh

Script.sh

```
for (( c=1; c<=5; c++ ))
do
adb shell input tap X Y
echo "已点击 $c 次"
sleep 5秒
完成
```

有关事件编号的完整列表

- 几个有趣事件的简短列表 [ADB Shell 输入事件](#)
- 参考文档
https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER.

Spaces are not accepted as the input, replace them with %s.

Send events

To simulate pressing the hardware power key

```
adb shell input keyevent 26
```

or alternatively

```
adb shell input keyevent POWER
```

Even if you don't have a hardware key you still can use a keyevent to perform the equivalent action

```
adb shell input keyevent CAMERA
```

Send touch event as input

```
adb shell input tap Xpoint Ypoint
```

Send swipe event as input

```
adb shell input swipe Xpoint1 Ypoint1 Xpoint2 Ypoint2 [DURATION*]
```

*DURATION is optional, default=300ms. [source](#)

Get X and Y points by enabling pointer location in developer option.

ADB sample shell script

To run a script in Ubuntu, Create script.sh right click the file and add read/write permission and tick **allow executing file as program**.

Open terminal emulator and run the command ./script.sh

Script.sh

```
for (( c=1; c<=5; c++ ))
do
adb shell input tap X Y
echo "Clicked $c times"
sleep 5s
done
```

For a comprehensive list of event numbers

- shortlist of several interesting events [ADB Shell Input Events](#)
- reference documentation
https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER.

第205.3节：列出包

打印所有包，或者仅打印包名中包含<FILTER>文本的包。

```
adb shell pm list packages [options] <FILTER>
```

所有 <FILTER>

```
adb shell pm list packages
```

属性：

-f 查看其关联的文件。

-i 查看软件包的安装程序。

-u 也包括未安装的软件包。

-u 也包括未安装的软件包。

筛选属性：

-d 用于禁用的软件包。

-e 用于启用的软件包。

-s 用于系统软件包。

-3 用于第三方软件包。

--user <USER_ID> 查询特定用户空间。

第205.4节：录制显示

版本 ≥ 4.4

录制运行 Android 4.4 (API 级别 19) 及更高版本设备的屏幕显示：

```
adb shell screenrecord [选项] <文件名>
adb shell screenrecord /sdcard/demo.mp4
```

(按 Ctrl-C 停止录制)

从设备下载文件：

```
adb pull /sdcard/demo.mp4
```

注意：通过按 Ctrl-C 停止屏幕录制，否则录制将在三分钟后或通过 --time-limit 设置的时间限制自动停止。

```
adb shell screenrecord --size <宽度x高度>
```

设置视频尺寸：1280x720。默认值为设备的原生显示分辨率（如果支持），否则为1280x720。为了获得最佳效果，请使用设备的高级视频编码（AVC）编码器支持的尺寸。

Section 205.3: List packages

Prints all packages, optionally only those whose package name contains the text in <FILTER>.

```
adb shell pm list packages [options] <FILTER>
```

All <FILTER>

```
adb shell pm list packages
```

Attributes:

-f to see their associated file.

-i See the installer for the packages.

-u to also include uninstalled packages.

-u Also include uninstalled packages.

Attributes that filter:

-d for disabled packages.

-e for enabled packages.

-s for system packages.

-3 for third party packages.

--user <USER_ID> for a specific user space to query.

Section 205.4: Recording the display

Version ≥ 4.4

Recording the display of devices running Android 4.4 (API level 19) and higher:

```
adb shell screenrecord [options] <filename>
adb shell screenrecord /sdcard/demo.mp4
```

(press Ctrl-C to stop recording)

Download the file from the device:

```
adb pull /sdcard/demo.mp4
```

Note: Stop the screen recording by pressing Ctrl-C, otherwise the recording stops automatically at three minutes or the time limit set by --time-limit.

```
adb shell screenrecord --size <WIDTHxHEIGHT>
```

Sets the video size: 1280x720. The default value is the device's native display resolution (if supported), 1280x720 if not. For best results, use a size supported by your device's Advanced Video Coding (AVC) encoder.

```
adb shell screenrecord --bit-rate <比特率>
```

设置视频的比特率，单位为兆比特每秒。默认值为4Mbps。您可以提高比特率以提升视频质量，但这会导致视频文件更大。以下示例将录制比特率设置为5Mbps：

```
adb shell screenrecord --bit-rate 0000000 /sdcard/demo.mp4
```

```
adb shell screenrecord --time-limit <TIME>
```

设置最大录制时间，单位为秒。默认和最大值为180秒（3分钟）。

```
adb shell screenrecord --rotate
```

将输出旋转90度。此功能为实验性功能。

```
adb shell screenrecord --verbose
```

在命令行屏幕上显示日志信息。如果不设置此选项，工具在运行时不会显示任何信息。

注意：这可能在某些设备上无法使用。

版本 < 4.4

屏幕录制命令不兼容4.4之前的Android版本

screenrecord 命令是一个用于录制运行 Android 4.4 (API 级别 19) 及更高版本设备显示屏的 shell 工具。该工具将屏幕活动录制为 MPEG-4 文件。

第205.5节：打开开发者选项

```
adb shell am start -n com.android.settings/.DevelopmentSettings
```

将导航您的设备/模拟器到开发者选项部分。

第205.6节：通过adb设置日期/时间

版本 ≥ 6.0

默认的SET格式是MMDDhhmm[[CC]YY][.ss]，（每项2位数字）

例如，要设置为7月17日10:10上午，且不更改当前年份，输入：

```
adb shell 'date 07171010.00'
```

提示1：日期更改不会立即生效，只有当系统时钟推进到下一分钟时，才会明显变化。

```
adb shell screenrecord --bit-rate <RATE>
```

Sets the video bit rate for the video, in megabits per second. The default value is 4Mbps. You can increase the bit rate to improve video quality, but doing so results in larger movie files. The following example sets the recording bit rate to 5Mbps:

```
adb shell screenrecord --bit-rate 5000000 /sdcard/demo.mp4
```

```
adb shell screenrecord --time-limit <TIME>
```

Sets the maximum recording time, in seconds. The default and maximum value is 180 (3 minutes).

```
adb shell screenrecord --rotate
```

Rotates the output 90 degrees. This feature is experimental.

```
adb shell screenrecord --verbose
```

Displays log information on the command-line screen. If you do not set this option, the utility does not display any information while running.

Note: This might not work on some devices.

Version < 4.4

The screen recording command isn't compatible with android versions pre 4.4

The screenrecord command is a shell utility for recording the display of devices running Android 4.4 (API level 19) and higher. The utility records screen activity to an MPEG-4 file.

Section 205.5: Open Developer Options

```
adb shell am start -n com.android.settings/.DevelopmentSettings
```

Will navigate your device/emulator to the Developer Options section.

Section 205.6: Set Date/Time via adb

Version ≥ 6.0

Default SET format is MMDDhhmm[[CC]YY][.ss], that's (2 digits each)

For example, to set July 17'th 10:10am, without changing the current year, type:

```
adb shell 'date 07171010.00'
```

Tip 1: the date change will not be reflected immediately, and a noticeable change will happen only after the system clock advances to the next minute.

您可以通过在调用中附加一个TIME_SET意图广播来强制更新，方法如下：

```
adb shell 'date 07171010.00 ; am broadcast -a android.intent.action.TIME_SET'
```

提示 2：同步 Android 时钟与本地机器：

Linux：

```
adb shell date `date +%m%d%H%M%S`
```

Windows (PowerShell)：

```
$currentDate = Get-Date -Format "MMddHHmmyyyy.ss" # Android首选格式  
adb shell "date $currentDate"
```

两个提示合并：

```
adb shell 'date `date +%m%d%H%M%S` ; am broadcast -a android.intent.action.TIME_SET'
```

版本 < 6.0

默认 SET 格式为 'YYYYMMDD.HHmmss'

```
adb shell 'date -s 20160117.095930'
```

提示：同步 Android 时钟与本地（基于 Linux）机器：

```
adb shell date -s `date +%G%m%d.%H%M%S`
```

第205.7节：生成“启动完成”广播

这适用于实现了BootListener的应用。通过杀死应用后使用以下命令测试您的应用：

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME -n  
your.package/your.app.BootListener
```

（将your.package/your.app.BootListener替换为正确的值）。

第205.8节：打印应用数据

此命令打印所有相关的应用数据：

- 版本代码
- 版本名称
- 已授予的权限 (Android API 23及以上)
- 等等..

```
adb shell dumpsys package <your.package.id>
```

第205.9节：使用chmod命令更改文件权限

注意，要更改文件权限，您的设备需要已获得root权限，su二进制文件不随出厂设备提供！

You can force an update by attaching a TIME_SET intent broadcast to your call, like that:

```
adb shell 'date 07171010.00 ; am broadcast -a android.intent.action.TIME_SET'
```

Tip 2: to synchronize Android's clock with your local machine:

Linux:

```
adb shell date `date +%m%d%H%M%S`
```

Windows (PowerShell):

```
$currentDate = Get-Date -Format "MMddHHmmyyyy.ss" # Android's preferred format  
adb shell "date $currentDate"
```

Both tips together:

```
adb shell 'date `date +%m%d%H%M%S` ; am broadcast -a android.intent.action.TIME_SET'
```

Version < 6.0

Default SET format is 'YYYYMMDD.HHmmss'

```
adb shell 'date -s 20160117.095930'
```

Tip: to synchronize Android's clock with your local (linux based) machine:

```
adb shell date -s `date +%G%m%d.%H%M%S`
```

Section 205.7: Generating a "Boot Complete" broadcast

This is relevant for apps that implement a BootListener. Test your app by killing your app and then test with:

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME -n  
your.package/your.app.BootListener
```

(replace your.package/your.app.BootListener with proper values).

Section 205.8: Print application data

This command print all relevant application data:

- version code
- version name
- granted permissions (Android API 23+)
- etc..

```
adb shell dumpsys package <your.package.id>
```

Section 205.9: Changing file permissions using chmod command

Notice, that in order to change file permissions, your device need to be rooted, su binary doesn't come with factory shipped devices!

约定：

```
adb shell su -c "chmod <数字权限> <文件>"
```

数字权限由用户、组和其他部分构成。

例如，如果您想将文件设置为所有人可读、可写和可执行，您的命令将是：

```
adb shell su -c "chmod 777 <文件路径>"
```

或者

```
adb shell su -c "chmod 000 <文件路径>"
```

如果您打算拒绝授予其任何权限。

第一位数字- 指定用户权限，第二位数字- 指定组权限，第三位数字- 指定其他用户权限 (world) 。

访问权限：

---	二进制值： 000, 八进制值： 0 (无权限)
--x	二进制值： 001, 八进制值： 1 (执行权限)
-w-	二进制值： 010, 八进制值： 2 (写权限)
-wx	二进制值： 011, 八进制值： 3 (写权限, 执行权限)
r--	二进制值： 100, 八进制值： 4 (读权限)
r-x	二进制值： 101, 八进制值： 5 (读权限, 执行权限)
rw-	二进制值： 110, 八进制值： 6 (读权限, 写权限)
rxw	二进制值： 111, 八进制值： 7 (读权限, 写权限, 执行权限)

Convention:

```
adb shell su -c "chmod <numeric-permission> <file>"
```

Numeric permission constructed from user, group and world sections.

For example, if you want to change file to be readable, writable and executable by everyone, this will be your command:

```
adb shell su -c "chmod 777 <file-path>"
```

Or

```
adb shell su -c "chmod 000 <file-path>"
```

if you intent to deny any permissions to it.

1st digit-specifies user permission, **2nd digit**- specifies group permission, **3rd digit** - specifies world (others) permission.

Access permissions:

---	binary value: 000, octal value: 0 (none)
--x	binary value: 001, octal value: 1 (execute)
-w-	binary value: 010, octal value: 2 (write)
-wx	binary value: 011, octal value: 3 (write, execute)
r--	binary value: 100, octal value: 4 (read)
r-x	binary value: 101, octal value: 5 (read, execute)
rw-	binary value: 110, octal value: 6 (read, write)
rxw	binary value: 111, octal value: 7 (read, write, execute)

第205.10节：查看外部/二级存储内容

查看内容：

```
adb shell ls \$EXTERNAL_STORAGE  
adb shell ls \$SECONDARY_STORAGE
```

查看路径：

```
adb shell echo \$EXTERNAL_STORAGE  
adb shell echo \$SECONDARY_STORAGE
```

第205.11节：在Android设备内终止进程

有时Android的logcat会无限运行，且错误来自某个非你所有的进程，导致电池消耗或使调试代码变得困难。

解决该问题的一个方便方法是在不重启设备的情况下，定位并终止引起问题的进程。

从Logcat中

```
03-10 11:41:40.010 1550-1627/? E/SomeProcess: ....
```

Section 205.10: View external/secondary storage content

View content:

```
adb shell ls \$EXTERNAL_STORAGE  
adb shell ls \$SECONDARY_STORAGE
```

View path:

```
adb shell echo \$EXTERNAL_STORAGE  
adb shell echo \$SECONDARY_STORAGE
```

Section 205.11: kill a process inside an Android device

Sometimes Android's logcat is running infinitely with errors coming from some process not own by you, draining battery or just making it hard to debug your code.

A convenient way to fix the problem without restarting the device is to locate and kill the process causing the problem.

From Logcat

```
03-10 11:41:40.010 1550-1627/? E/SomeProcess: ....
```

注意进程号：1550

现在我们可以打开shell并终止该进程。注意我们无法终止root进程。

```
adb shell
```

在shell内部，我们可以使用以下命令进一步检查该进程

```
ps -x | grep 1550
```

如果需要，我们可以终止它：

```
kill -9 1550
```

notice the process number: 1550

Now we can open a shell and kill the process. Note that we cannot kill root process.

```
adb shell
```

inside the shell we can check more about the process using

```
ps -x | grep 1550
```

and kill it if we want:

```
kill -9 1550
```

第206章：Ping ICMP

可以通过创建一个新进程来运行ping请求，从而在Android中执行ICMP Ping请求。可以在该进程完成ping请求后评估请求的结果。

第206.1节：执行单次Ping

此示例尝试执行一次Ping请求。runtime.exec方法调用中的ping命令可以修改为任何你在命令行中可能执行的有效ping命令。

```
try {
    Process ipProcess = runtime.exec("/system/bin/ping -c 1 8.8.8.8");
    int exitValue = ipProcess.waitFor();
    ipProcess.destroy();

    if(exitValue == 0){
        // 成功
    } else {
        // 失败
    }
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
```

Chapter 206: Ping ICMP

The ICMP Ping request can be performed in Android by creating a new process to run the ping request. The outcome of the request can be evaluated upon the completion of the ping request from within its process.

Section 206.1: Performs a single Ping

This example attempts a single Ping request. The ping command inside the `runtime.exec` method call can be modified to any valid ping command you might perform yourself in the command line.

```
try {
    Process ipProcess = runtime.exec("/system/bin/ping -c 1 8.8.8.8");
    int exitValue = ipProcess.waitFor();
    ipProcess.destroy();

    if(exitValue == 0){
        // Success
    } else {
        // Failure
    }
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
```

第207章：AIDL

AIDL是Android接口定义语言。

什么？为什么？怎么做？

什么？它是一种绑定服务。该AIDL服务将一直保持活动状态，直到至少有一个客户端存在。它基于封送和解封送的概念工作。

为什么？远程应用程序可以访问您的服务 + 多线程。（远程应用程序请求）。

如何？创建 .aidl 文件 实现接口 向客户端暴露接口

第207.1节：AIDL 服务

ICalculator.aidl

```
// 在此处使用 import 语句声明任何非默认类型
```

```
interface ICalculator {
    int add(int x,int y);
    int sub(int x,int y);
}
```

AidlService.java

```
public class AidlService extends Service {

    private static final String TAG = "AIDLServiceLogs";
    private static final String className = " AidlService";

    public AidlService() {
        Log.i(TAG, className+" Constructor");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: 返回与服务通信的通道。
        Log.i(TAG, className+" onBind");
        return iCalculator.asBinder();
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, className+" onCreate");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, className+" onDestroy");
    }
}
```

Chapter 207: AIDL

AIDL is Android interface definition language.

What? Why? How ?

What? It is a bounded services. This AIDL service will be active till atleast one of the client is exist. It works based on marshaling and unmarshaling concept.

Why? Remote applications can access your service + Multi Threading.(Remote application request).

How? Create the .aidl file Implement the interface Expose the interface to clients

Section 207.1: AIDL Service

ICalculator.aidl

```
// Declare any non-default types here with import statements
```

```
interface ICalculator {
    int add(int x,int y);
    int sub(int x,int y);
}
```

AidlService.java

```
public class AidlService extends Service {

    private static final String TAG = "AIDLServiceLogs";
    private static final String className = " AidlService";

    public AidlService() {
        Log.i(TAG, className+" Constructor");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        Log.i(TAG, className+" onBind");
        return iCalculator.asBinder();
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, className+" onCreate");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, className+" onDestroy");
    }
}
```

```

ICalculator.Stub iCalculator = new ICalculator.Stub() {
    @Override
    public int add(int x, int y) throws RemoteException {
        Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
        int z = x+y;
        return z;
    }

    @Override
    public int sub(int x, int y) throws RemoteException {
        Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
        int z = x-y;
        return z;
    }
};

```

服务连接

```

// 返回存根作为接口
ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        Log.i(TAG, className + " onServiceConnected");
        iCalculator = ICalculator.Stub.asInterface(service);
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {

        unbindService(serviceConnection);
    }
};

```

```

ICalculator.Stub iCalculator = new ICalculator.Stub() {
    @Override
    public int add(int x, int y) throws RemoteException {
        Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
        int z = x+y;
        return z;
    }

    @Override
    public int sub(int x, int y) throws RemoteException {
        Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
        int z = x-y;
        return z;
    }
};

```

Service Connection

```

// Return the stub as interface
ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        Log.i(TAG, className + " onServiceConnected");
        iCalculator = ICalculator.Stub.asInterface(service);
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {

        unbindService(serviceConnection);
    }
};

```

第208章：Android游戏开发

使用Java在Android平台上创建游戏的简短介绍

第208.1节：使用Canvas和SurfaceView的游戏

本节介绍如何使用SurfaceView创建一个基本的2D游戏。

首先，我们需要一个活动：

```
public class GameLauncher extends AppCompatActivity {  
  
    private Game game;  
    @Override  
    public void onCreate(Bundle sis){  
        super.onCreate(sis);  
        game = new Game(GameLauncher.this); //初始化游戏实例  
        setContentView(game); //将内容视图设置为游戏的SurfaceView  
        //也可以使用自定义的XML文件，然后通过findViewById获取游戏实例。  
    }  
}
```

该活动还必须在Android清单文件中声明。

现在开始游戏本身。首先，我们实现一个游戏线程：

```
public class Game extends SurfaceView implements SurfaceHolder.Callback, Runnable{  
  
    /**  
     * 持有Surface帧  
     */  
    private SurfaceHolder holder;  
  
    /**  
     * 绘制线程  
     */  
    private Thread drawThread;  
  
    /**  
     * 当表面准备好绘制时为真  
     */  
    private boolean surfaceReady = false;  
  
    /**  
     * 绘制线程标志  
     */  
    private boolean drawingActive = false;  
  
    /**  
     * 60帧每秒的每帧时间  
     */  
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);  
  
    private static final String LOGTAG = "surface";  
}
```

Chapter 208: Android game development

A short introduction to creating a game on the Android platform using Java

Section 208.1: Game using Canvas and SurfaceView

This covers how you can create a basic 2D game using SurfaceView.

First, we need an activity:

```
public class GameLauncher extends AppCompatActivity {  
  
    private Game game;  
    @Override  
    public void onCreate(Bundle sis){  
        super.onCreate(sis);  
        game = new Game(GameLauncher.this); //Initialize the game instance  
        setContentView(game); //setContentView to the game surfaceview  
        //Custom XML files can also be used, and then retrieve the game instance using findViewById.  
    }  
}
```

The activity also has to be declared in the Android Manifest.

Now for the game itself. First, we start by implementing a game thread:

```
public class Game extends SurfaceView implements SurfaceHolder.Callback, Runnable{  
  
    /**  
     * Holds the surface frame  
     */  
    private SurfaceHolder holder;  
  
    /**  
     * Draw thread  
     */  
    private Thread drawThread;  
  
    /**  
     * True when the surface is ready to draw  
     */  
    private boolean surfaceReady = false;  
  
    /**  
     * Drawing thread flag  
     */  
    private boolean drawingActive = false;  
  
    /**  
     * Time per frame for 60 FPS  
     */  
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);  
  
    private static final String LOGTAG = "surface";  
}
```

* 所有构造函数均被重写，以确保无论通过XML文件还是编程方式使用不同的构造函数时功能正常

```
/*
public Game(Context context) {
    super(context);
init();
}
public 游戏(上下文 context, 属性集 attrs) {
    父类(context, attrs);
init();
}
public 游戏(上下文 context, 属性集 attrs, 整数 defStyleAttr) {
    父类(context, attrs, defStyleAttr);
init();
}
@TargetApi(21)
public 游戏(上下文 context, 属性集 attrs, 整数 defStyleAttr, 整数 defStyleRes) {
    父类(context, attrs, defStyleAttr, defStyleRes);
init();
}

public 无效 初始化(上下文 c) {
    this.c = c;

SurfaceHolder 持有者 = 获取持有者();
    持有者.添加回调(this);
设置可聚焦(真);
    //稍后在这里初始化其他内容
}

public 无效 渲染(画布 c){
    //游戏渲染在这里
}

public 无效 计时(){
    //游戏逻辑在这里
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
    if (width == 0 || height == 0){
        return;
    }

    // 调整你的界面大小
}

@Override
public void surfaceCreated(SurfaceHolder holder){
    this.holder = holder;

    if (drawThread != null){
Log.d(LOGTAG, "绘图线程仍在运行中..");
        drawingActive = false;
        try{
drawThread.join();
            } catch (InterruptedException e){}
    }

surfaceReady = true;
    startDrawThread();
}
```

* All the constructors are overridden to ensure functionality if one of the different constructors are used through an XML file or programmatically

```
/*
public Game(Context context) {
    super(context);
init();
}
public Game(Context context, AttributeSet attrs) {
    super(context, attrs);
init();
}
public Game(Context context, AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);
init();
}
@TargetApi(21)
public Game(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
    super(context, attrs, defStyleAttr, defStyleRes);
init();
}

public void init(Context c) {
    this.c = c;

SurfaceHolder holder = getHolder();
holder.addCallback(this);
setFocusable(true);
//Initialize other stuff here later
}

public void render(Canvas c){
    //Game rendering here
}

public void tick(){
    //Game logic here
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
    if (width == 0 || height == 0){
        return;
    }

    // resize your UI
}

@Override
public void surfaceCreated(SurfaceHolder holder){
    this.holder = holder;

    if (drawThread != null){
        Log.d(LOGTAG, "draw thread still active..");
        drawingActive = false;
        try{
            drawThread.join();
        } catch (InterruptedException e){}
    }

surfaceReady = true;
    startDrawThread();
}
```

```

Log.d(LOGTAG, "已创建");
}

@Override
public void surfaceDestroyed(SurfaceHolder holder){
    // 画面不再使用 - 停止绘图线程
    stopDrawThread();
    // 并释放画面
    holder.getSurface().release();

    this.holder = null;
    surfaceReady = false;
    Log.d(LOGTAG, "Destroyed");
}

@Override
public boolean onTouchEvent(MotionEvent event){
    // 处理触摸事件
    return true;
}

/**
 * 停止绘图线程
 */
public void stopDrawThread(){
    if (drawThread == null){
        Log.d(LOGTAG, "DrawThread is null");
        return;
    }
    drawingActive = false;
    while (true){
        try{
Log.d(LOGTAG, "请求最后一帧");
        drawThread.join(5000);
        break;
    } 捕获 (异常 e) {
Log.e(LOGTAG, "无法与绘图线程合并");
    }
}
drawThread = null;
}

/**
 * 创建一个新的绘制线程并启动它。
 */
public void startDrawThread(){
    if (surfaceReady && drawThread == null){
        drawThread = new Thread(this, "绘制线程");
        drawingActive = true;
drawThread.start();
    }
}

@Override
public void run() {
Log.d(LOGTAG, "绘制线程已启动");
    long frameStartTime;
    long frameTime;

    /*
    * 为了在 Nexus 7 上可靠运行，我们在绘制线程开始时加入约 500 毫秒的延迟
    * (AOSP - 问题 58385)
    */
}

```

```

Log.d(LOGTAG, "Created");
}

@Override
public void surfaceDestroyed(SurfaceHolder holder){
    // Surface is not used anymore - stop the drawing thread
    stopDrawThread();
    // and release the surface
    holder.getSurface().release();

    this.holder = null;
    surfaceReady = false;
    Log.d(LOGTAG, "Destroyed");
}

@Override
public boolean onTouchEvent(MotionEvent event){
    // Handle touch events
    return true;
}

/**
 * Stops the drawing thread
 */
public void stopDrawThread(){
    if (drawThread == null){
        Log.d(LOGTAG, "DrawThread is null");
        return;
    }
    drawingActive = false;
    while (true){
        try{
            Log.d(LOGTAG, "Request last frame");
            drawThread.join(5000);
            break;
        } catch (Exception e) {
            Log.e(LOGTAG, "Could not join with draw thread");
        }
    }
    drawThread = null;
}

/**
 * Creates a new draw thread and starts it.
 */
public void startDrawThread(){
    if (surfaceReady && drawThread == null){
        drawThread = new Thread(this, "Draw thread");
        drawingActive = true;
        drawThread.start();
    }
}

@Override
public void run() {
    Log.d(LOGTAG, "Draw thread started");
    long frameStartTime;
    long frameTime;

    /*
    * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing thread
    * (AOSP - Issue 58385)
    */
}

```

```

/*
if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
    android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
    android.os.Build.MODEL.equalsIgnoreCase("Nexus 7")) {
    Log.w(LOGTAG, "休眠 500 毫秒 (设备：华硕 Nexus 7)");
    try {
        Thread.sleep(500);
    } catch (InterruptedException ignored) {}
}

while (drawing) {
    if (sf == null) {
        return;
    }

frameStartTime = System.nanoTime();
Canvas canvas = sf.lockCanvas();
if (canvas != null) {
    try {
        synchronized (sf) {
            tick();
            render(canvas);
        }
    } finally {
        sf.unlockCanvasAndPost(canvas);
    }
}

// 计算绘制帧所需的时间，单位为毫秒
frameTime = (System.nanoTime() - frameStartTime) / 0000000;

if (frameTime < MAX_FRAME_TIME){
    try {
        Thread.sleep(MAX_FRAME_TIME - frameTime);
    } catch (InterruptedException e) {
        // 忽略
    }
}

Log.d(LOGTAG, "绘制线程已结束");
}
}

```

这是基本部分。现在你已经具备了在屏幕上绘图的能力。

现在，让我们开始添加两个整数：

```

public final int x = 100;// 这个是静态的原因将在游戏可运行时展示
public int y;
public int velY;

```

接下来这部分，你需要一张图片。它应该大约是100x100，但也可以更大或更小。对于学习来说，也可以使用矩形(Rect)（但这需要稍微修改下面的代码）

现在，我们声明一个Bitmap：

```

private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);

```

```

/*
if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
    android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
    android.os.Build.MODEL.equalsIgnoreCase("Nexus 7")) {
    Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
    try {
        Thread.sleep(500);
    } catch (InterruptedException ignored) {}
}

while (drawing) {
    if (sf == null) {
        return;
    }

frameStartTime = System.nanoTime();
Canvas canvas = sf.lockCanvas();
if (canvas != null) {
    try {
        synchronized (sf) {
            tick();
            render(canvas);
        }
    } finally {
        sf.unlockCanvasAndPost(canvas);
    }
}

// calculate the time required to draw the frame in ms
frameTime = (System.nanoTime() - frameStartTime) / 1000000;

if (frameTime < MAX_FRAME_TIME){
    try {
        Thread.sleep(MAX_FRAME_TIME - frameTime);
    } catch (InterruptedException e) {
        // ignore
    }
}

Log.d(LOGTAG, "Draw thread finished");
}
}

```

That is the basic part. Now you have the ability to draw onto the screen.

Now, let's start by adding to integers:

```

public final int x = 100;//The reason for this being static will be shown when the game is runnable
public int y;
public int velY;

```

For this next part, you are going to need an image. It should be about 100x100 but it can be bigger or smaller. For learning, a Rect can also be used (but that requires change in code a little bit down)

Now, we declare a Bitmap:

```

private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);

```

在渲染(render)中，我们需要绘制这个位图。

```
...  
c.drawBitmap(PLAYER_BMP, x, y, null);  
...
```

在启动之前 还有一些事情要做

我们首先需要一个布尔值：

```
boolean up = false;
```

在 onTouchEvent 中，我们添加：

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){  
    up = true;  
}else if(ev.getAction() == MotionEvent.ACTION_UP){  
    up = false;  
}
```

在 tick 中我们需要这个来移动玩家：

```
if(up){  
    velY -=1;  
}  
else{  
    velY +=1;  
}  
if(velY >14)velY = 14;  
if(velY <-14)velY = -14;  
y += velY *2;
```

现在我们需要在init中添加这些内容：

```
WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);  
Display display = wm.getDefaultDisplay();  
Point size = new Point();  
display.getSize(size);  
WIDTH = size.x;  
HEIGHT = size.y;  
y = HEIGHT/ 2 - PLAYER_BMP.getHeight();
```

我们还需要这些变量：

```
public static int WIDTH, HEIGHT;
```

此时，游戏可以运行了。也就是说，你可以启动它并进行测试。

现在你应该有一个玩家图像或矩形在屏幕上上下移动。如果需要，玩家可以被创建为一个自定义类。然后所有与玩家相关的内容都可以移到该类中，并使用该类的实例来移动、渲染和执行其他逻辑。

现在，正如你可能在测试中看到的那样，它会飞出屏幕。所以我们需要限制它。

首先，我们需要声明一个矩形（Rect）：

In render, we need to draw this bitmap.

```
...  
c.drawBitmap(PLAYER_BMP, x, y, null);  
...
```

BEFORE LAUNCHING there are still some things to be done

We need a boolean first:

```
boolean up = false;
```

in onTouchEvent, we add:

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){  
    up = true;  
}else if(ev.getAction() == MotionEvent.ACTION_UP){  
    up = false;  
}
```

And in tick we need this to move the player:

```
if(up){  
    velY -=1;  
}  
else{  
    velY +=1;  
}  
if(velY >14)velY = 14;  
if(velY <-14)velY = -14;  
y += velY *2;
```

and now we need this in init:

```
WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);  
Display display = wm.getDefaultDisplay();  
Point size = new Point();  
display.getSize(size);  
WIDTH = size.x;  
HEIGHT = size.y;  
y = HEIGHT/ 2 - PLAYER_BMP.getHeight();
```

And we need these to variables:

```
public static int WIDTH, HEIGHT;
```

At this point, the game is runnable. Meaning you can launch it and test it.

Now you should have a player image or rect going up and down the screen. The player can be created as a custom class if needed. Then all the player-related things can be moved into that class, and use an instance of that class to move, render and do other logic.

Now, as you probably saw under testing it flies off the screen. So we need to limit it.

First, we need to declare the Rect:

```
private Rect screen;
```

在初始化中，在初始化宽度和高度之后，我们创建一个新的矩形作为屏幕。

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

现在我们需要另一个以方法形式存在的矩形：

```
private Rect getPlayerBound(){  
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight());  
}
```

在 tick 方法中：

```
if(!getPlayerBound().intersects(screen)){  
    gameOver = true;  
}
```

gameOver 的实现也可以用来显示游戏的开始。

游戏的其他值得注意的方面：

保存（当前文档中缺失）

```
private Rect screen;
```

In init, after initializing width and height, we create a new rect that is the screen.

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

Now we need another rect in the form of a method:

```
private Rect getPlayerBound(){  
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight());  
}
```

and in tick:

```
if(!getPlayerBound().intersects(screen)){  
    gameOver = true;  
}
```

The implementation of gameOver can also be used to show the start of a game.

Other aspects of a game worth noting:

Saving(currently missing in documentation)

第209章：使用 Kotlin 进行 Android 编程

在 Android Studio 中使用 Kotlin 是一项简单的任务，因为 Kotlin 是由 JetBrains 开发的。JetBrains 也是 IntelliJ IDEA 的开发公司——Android Studio 的基础集成开发环境（IDE）。这就是为什么几乎不存在兼容性问题的原因。

第209.1节：安装 Kotlin 插件

首先，你需要安装 Kotlin 插件。

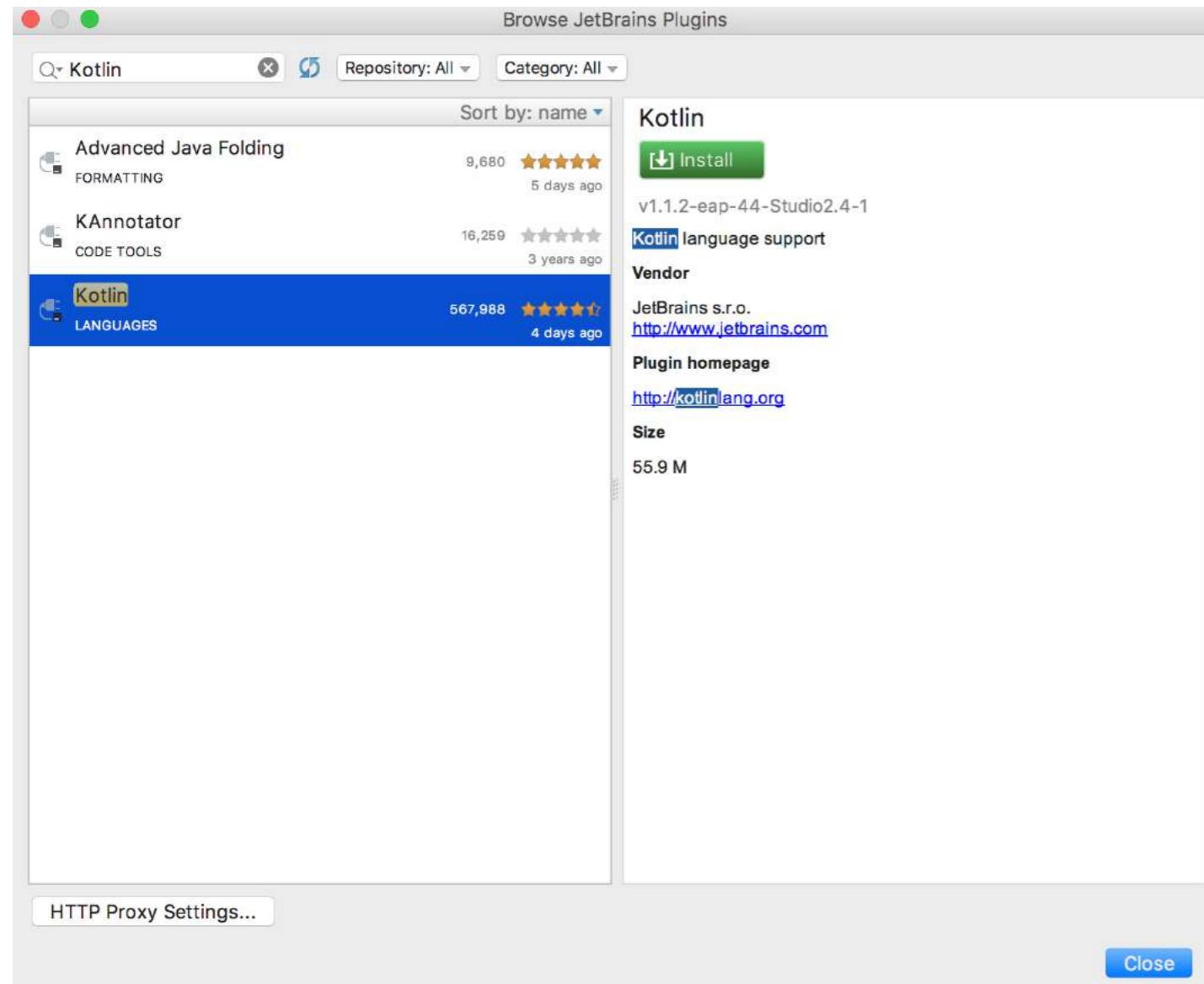
对于 Windows：

- 导航至 文件 → 设置 → 插件 → 安装 JetBrains 插件

对于 Mac：

- 导航至 Android Studio → 偏好设置 → 插件 → 安装 JetBrains 插件

然后搜索并安装 Kotlin。完成后需要重启 IDE。



Chapter 209: Android programming with Kotlin

Using Kotlin with Android Studio is an easy task as Kotlin is developed by JetBrains. It is the same company that stands behind IntelliJ IDEA - a base IDE for Android Studio. That is why there are almost none problems with the compatibility.

Section 209.1: Installing the Kotlin plugin

First, you'll need to install the Kotlin plugin.

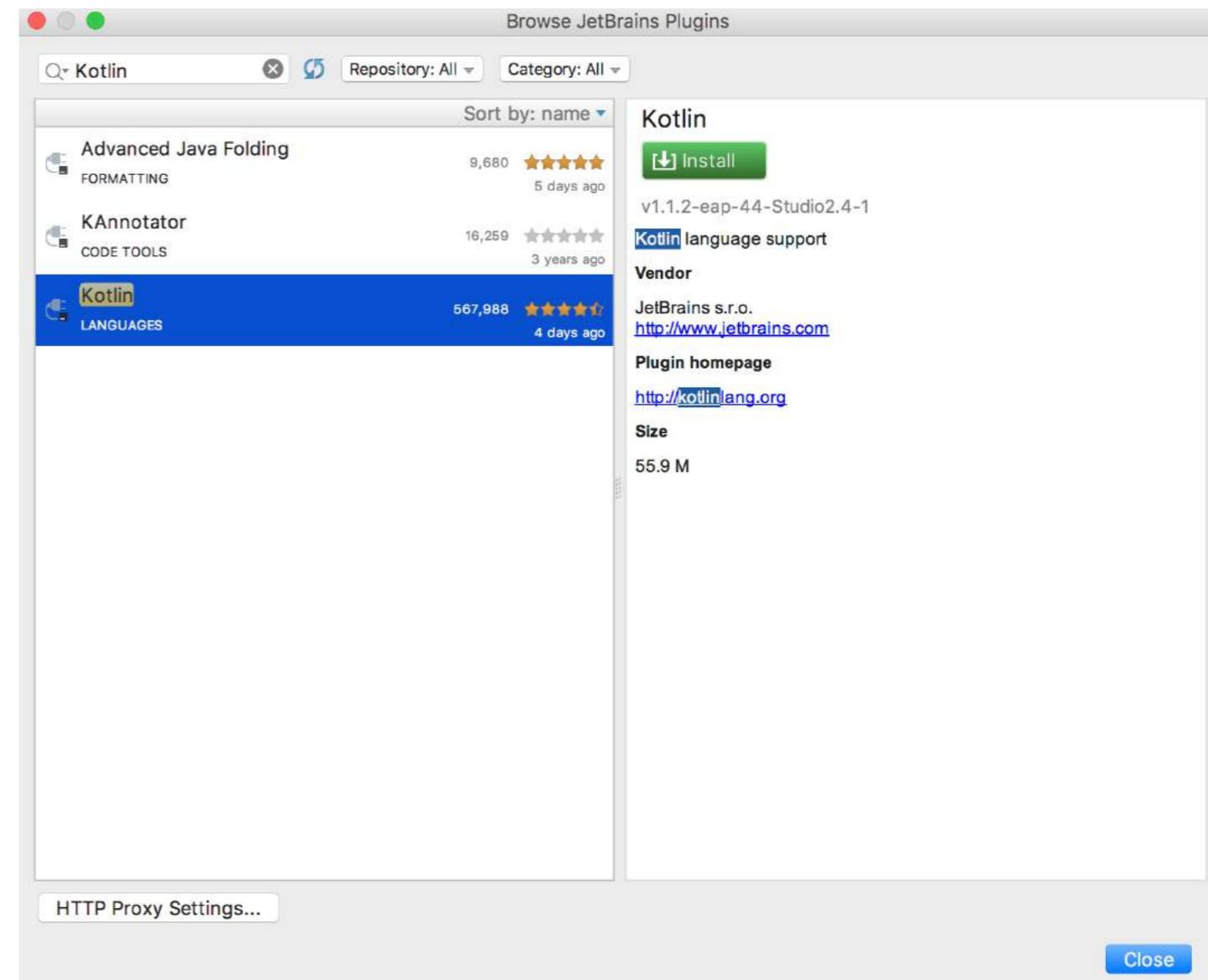
For Windows:

- Navigate to [File](#) → [Settings](#) → [Plugins](#) → [Install JetBrains plugin](#)

For Mac:

- Navigate to [Android Studio](#) → [Preferences](#) → [Plugins](#) → [Install JetBrains plugin](#)

And then search for and install Kotlin. You'll need to restart the IDE after this completes.



第 209.2 节：使用

Kotlin 配置现有的 Gradle 项目

你可以在 Android Studio 中创建一个新项目，然后为其添加 Kotlin 支持，或者修改你现有的项目。要做到这一点，你需要：

1. 向根 gradle 文件添加依赖 - 你需要为kotlin-android插件添加依赖
根build.gradle文件。

```
buildscript {  
  
    repositories {  
        jcenter()  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:2.3.1'  
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.1.2'  
    }  
}  
  
allprojects {  
    repositories {  
        jcenter()  
    }  
}  
  
task clean(类型: 删除) {  
    删除 rootProject.buildDir  
}
```

2. 应用 Kotlin Android 插件 - 只需在模块的 build.gradle 文件中添加 apply plugin: 'kotlin-android'。
3. 添加对 Kotlin 标准库的依赖 - 在依赖部分添加 'org.jetbrains.kotlin:kotlin-stdlib:1.1.2' 到模块的 build.gradle 文件的依赖部分。

对于新项目，build.gradle 文件可能如下所示：

```
apply plugin: 'com.android.application'  
apply plugin: 'kotlin-android'  
  
android {  
    compileSdkVersion 25  
    buildToolsVersion "25.0.2"  
    defaultConfig {  
        applicationId "org.example.example"  
        minSdkVersion 16  
        targetSdkVersion 25  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
    }  
}  
  
dependencies {
```

Section 209.2: Configuring an existing Gradle project with Kotlin

You can create a New Project in Android Studio and then add Kotlin support to it or modify your existing project. To do it, you have to:

1. **Add dependency to a root gradle file** - you have to add the dependency for kotlin-android plugin to a root build.gradle file.

```
buildscript {  
  
    repositories {  
        jcenter()  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:2.3.1'  
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.1.2'  
    }  
}  
  
allprojects {  
    repositories {  
        jcenter()  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```

2. **Apply Kotlin Android Plugin** - simply add apply plugin: 'kotlin-android' to a module build.gradle file.
3. **Add dependency to Kotlin stdlib** - add the dependency to 'org.jetbrains.kotlin:kotlin-stdlib:1.1.2' to the dependency section in a module build.gradle file.

For a new project, build.gradle file could looks like this:

```
apply plugin: 'com.android.application'  
apply plugin: 'kotlin-android'  
  
android {  
    compileSdkVersion 25  
    buildToolsVersion "25.0.2"  
    defaultConfig {  
        applicationId "org.example.example"  
        minSdkVersion 16  
        targetSdkVersion 25  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
    }  
}  
  
dependencies {
```

```

compile 'org.jetbrains.kotlin:kotlin-stdlib:1.1.1'
compile 'com.android.support.constraint:constraint-layout:1.0.2'
compile 'com.android.support:appcompat-v7:25.3.1'

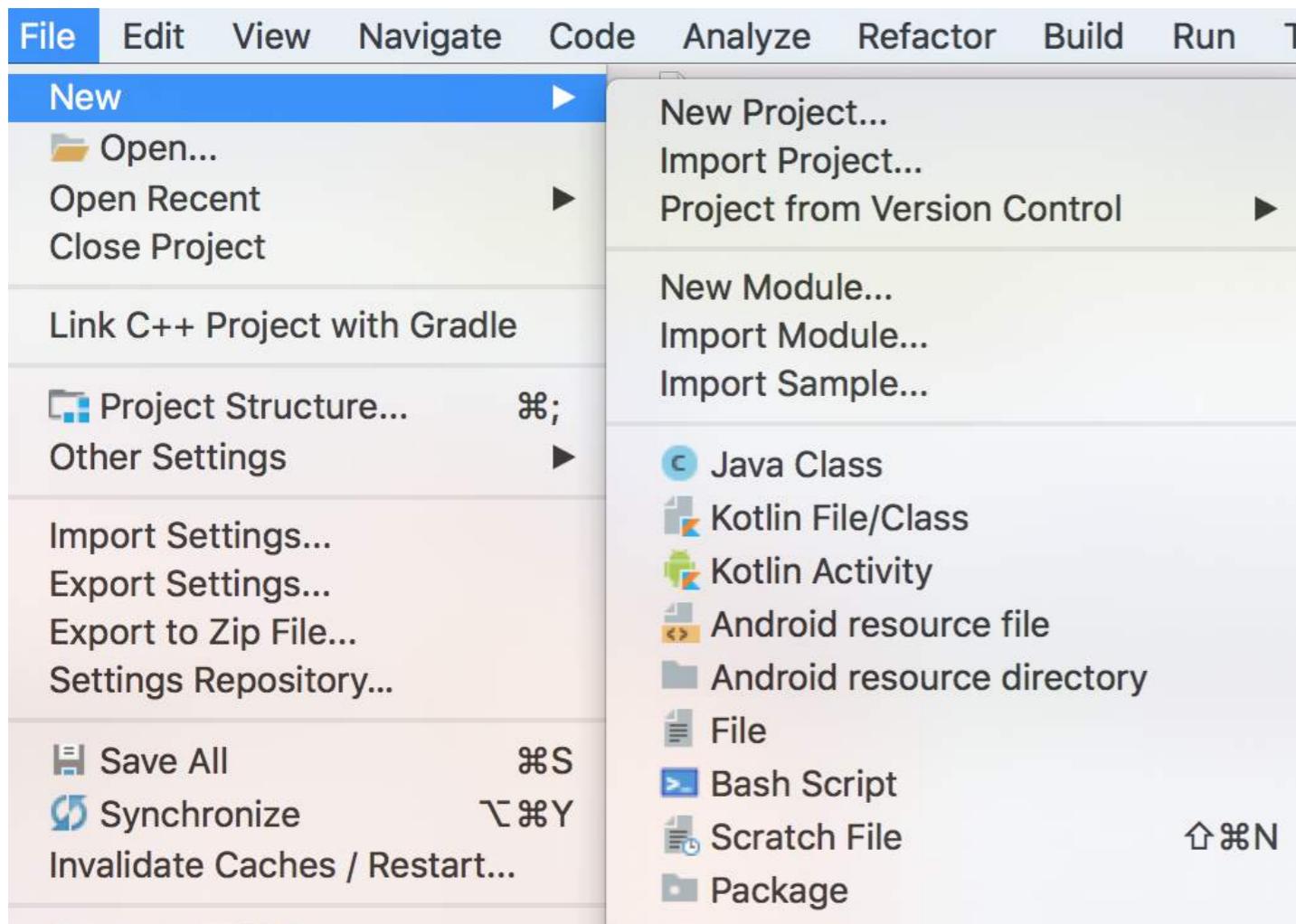
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
    exclude group: 'com.android.support', module: 'support-annotations'
})

testCompile 'junit:junit:4.12'
}

```

第209.3节：创建新的Kotlin活动

1. 点击 文件 → 新建 → Kotlin活动。
2. 选择活动的类型。
3. 选择活动的名称和其他参数。
4. 完成。



最终的类可能如下所示：

```

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}

```

```

compile 'org.jetbrains.kotlin:kotlin-stdlib:1.1.1'
compile 'com.android.support.constraint:constraint-layout:1.0.2'
compile 'com.android.support:appcompat-v7:25.3.1'

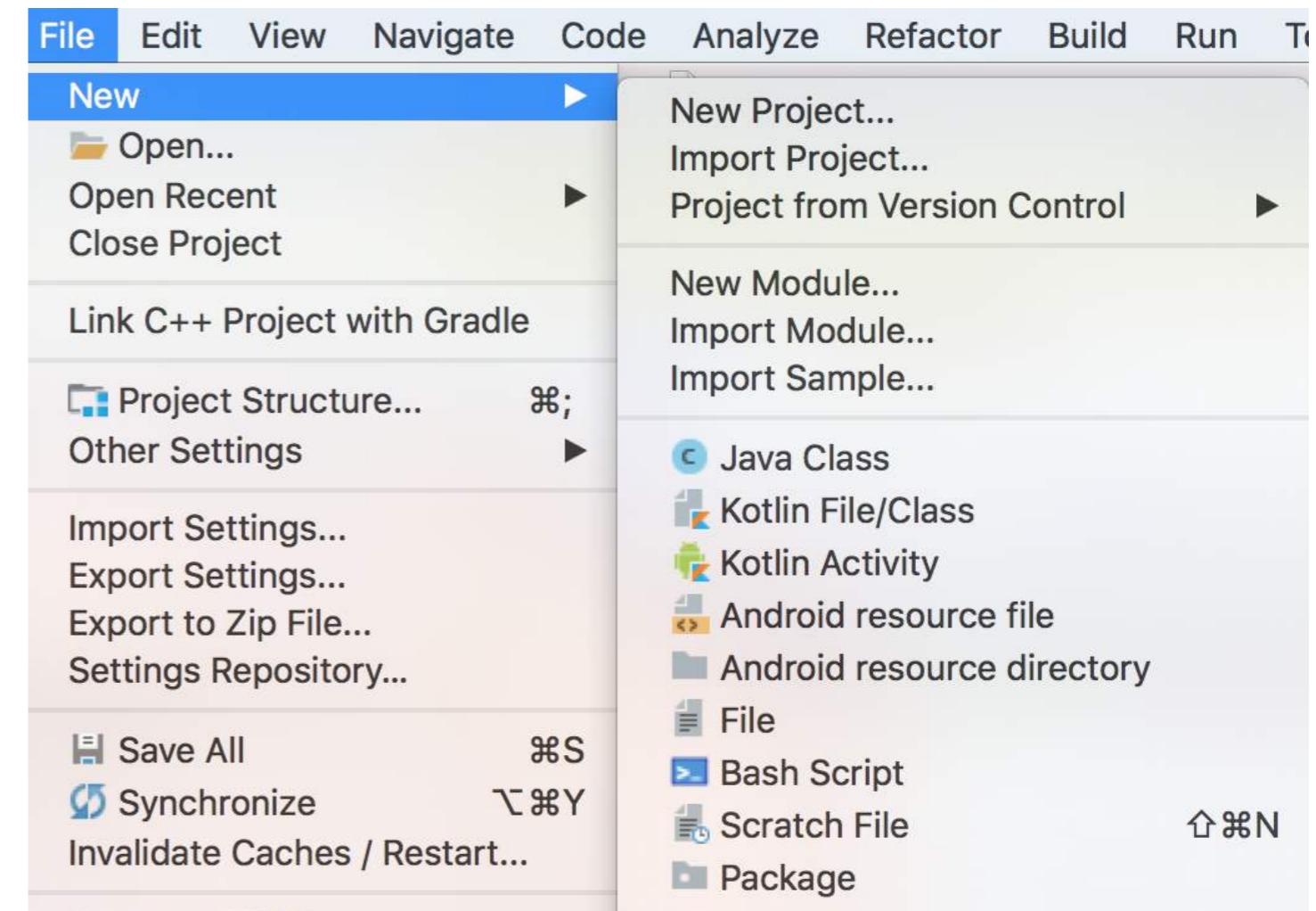
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
    exclude group: 'com.android.support', module: 'support-annotations'
})

testCompile 'junit:junit:4.12'
}

```

Section 209.3: Creating a new Kotlin Activity

1. Click to [File](#) → [New](#) → [Kotlin Activity](#).
2. Choose a type of the Activity.
3. Select name and other parameter for the Activity.
4. Finish.



Final class could look like this:

```

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}

```

```
}
```

第209.4节：将现有Java代码转换为Kotlin

Kotlin插件支持Android Studio将现有Java文件转换为Kotlin文件。选择一个Java文件并调用操作“将Java文件转换为Kotlin文件”：

```
public class MainActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
    Enter action or option name:  Include non-menu actions (⇧⌘A)  
     Convert Java F Kotlin  
    @Override  
    public // Convert Java File to Kotlin File (⌃⇧⌘J) Code  
    getMenuInflater().inflate(R.menu.menu_main, menu);  
    return true;  
}
```

```
}
```

Section 209.4: Converting existing Java code to Kotlin

Kotlin Plugin for Android Studio support converting existing Java files to Kotlin files. Choose a Java file and invoke action Convert Java File to Kotlin File:

```
public class MainActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
    Enter action or option name:  Include non-menu actions (⇧⌘A)  
     Convert Java F Kotlin  
    @Override  
    public // Convert Java File to Kotlin File (⌃⇧⌘J) Code  
    getMenuInflater().inflate(R.menu.menu_main, menu);  
    return true;  
}
```

第209.5节：启动一个新的Activity

```
fun startNewActivity(){  
    val intent: Intent = Intent(context, Activity::class.java)  
    startActivity(intent)  
}
```

你可以像在Java中一样向intent添加额外信息。

```
fun startNewActivityWithIntents(){  
    val intent: Intent = Intent(context, Activity::class.java)  
    intent.putExtra(KEY_NAME, KEY_VALUE)  
    startActivity(intent)  
}
```

```
}
```

Section 209.5: Starting a new Activity

```
fun startNewActivity(){  
    val intent: Intent = Intent(context, Activity::class.java)  
    startActivity(intent)  
}
```

You can add extras to the intent just like in Java.

```
fun startNewActivityWithIntents(){  
    val intent: Intent = Intent(context, Activity::class.java)  
    intent.putExtra(KEY_NAME, KEY_VALUE)  
    startActivity(intent)  
}
```

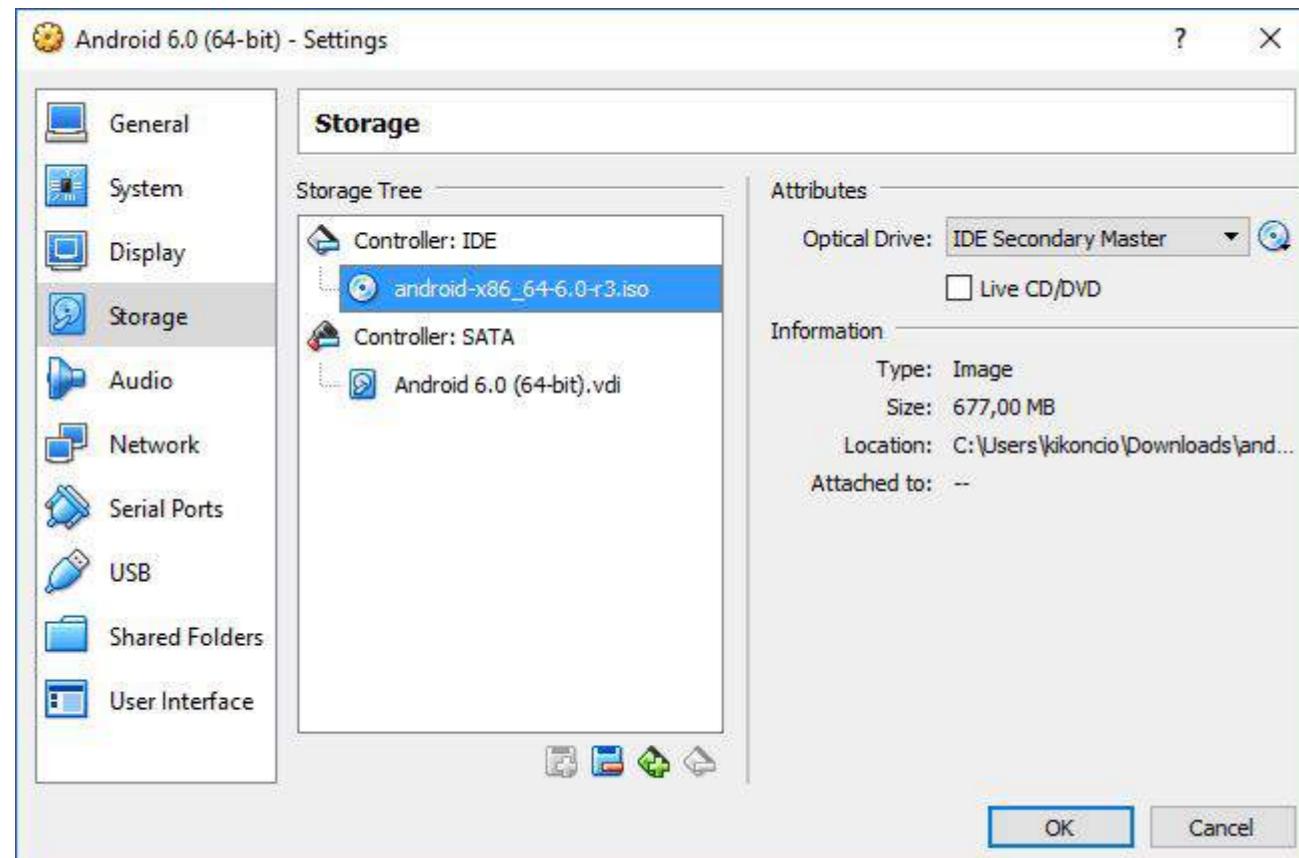
第210章：VirtualBox中的Android-x86

本节的目的是介绍如何安装和使用带有Android-x86的VirtualBox进行调试。这是一项困难的任务，因为不同版本之间存在差异。目前我将介绍我使用过的6.0版本，然后我们需要寻找相似之处。

本节不详细介绍VirtualBox或Linux，但展示了我用来使其工作的命令。

第210.1节：支持SDCARD的虚拟硬盘设置

使用刚创建的虚拟硬盘，从光驱中加载android-x86镜像启动虚拟机。



启动后，您可以看到Live CD的grub菜单

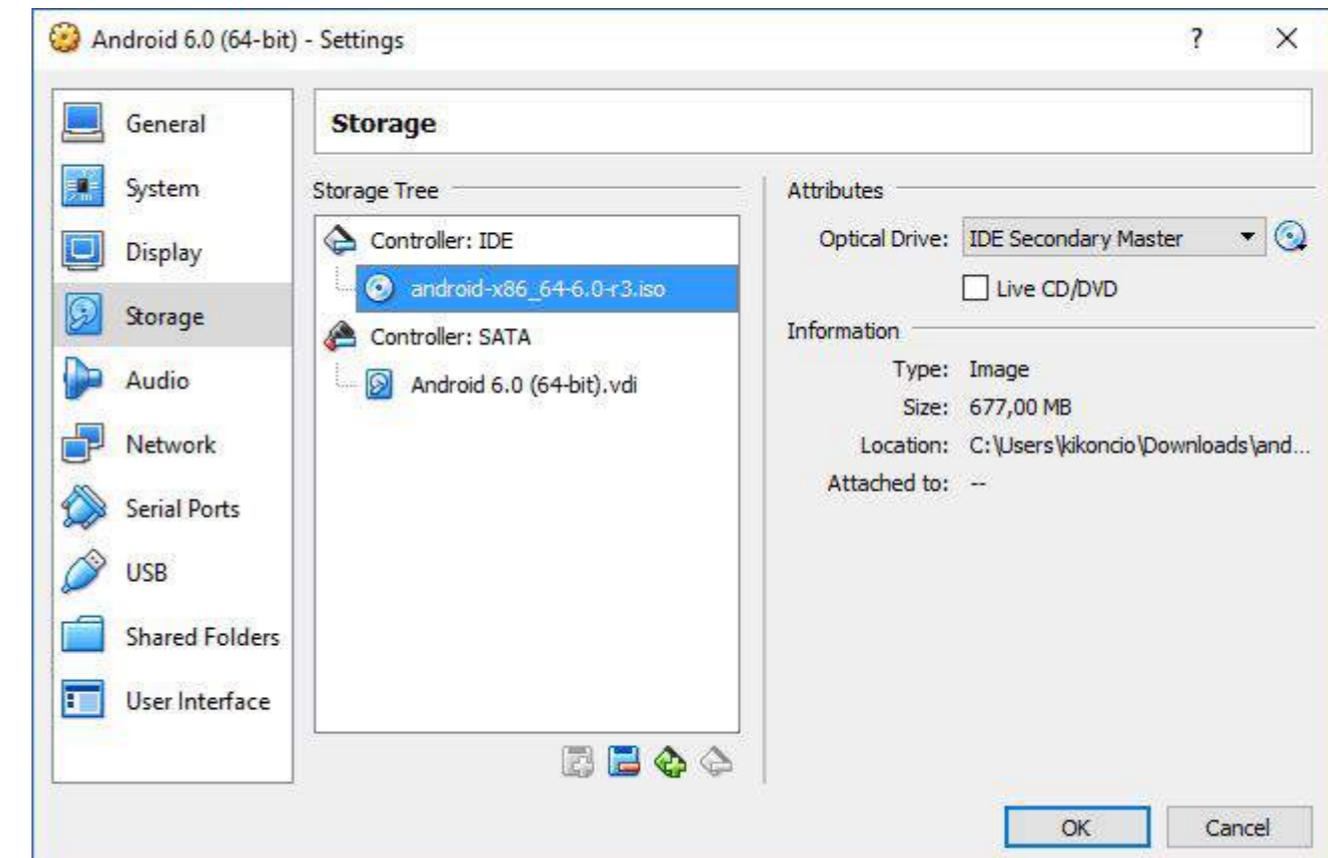
Chapter 210: Android-x86 in VirtualBox

The idea of this section is to cover how to install and use the VirtualBox with Android-x86 for debugging purposes. This is a difficult task because there are differences between versions. For the moment I'm going to cover 6.0 which is the one that I had to work with and then we'll have to find similarities.

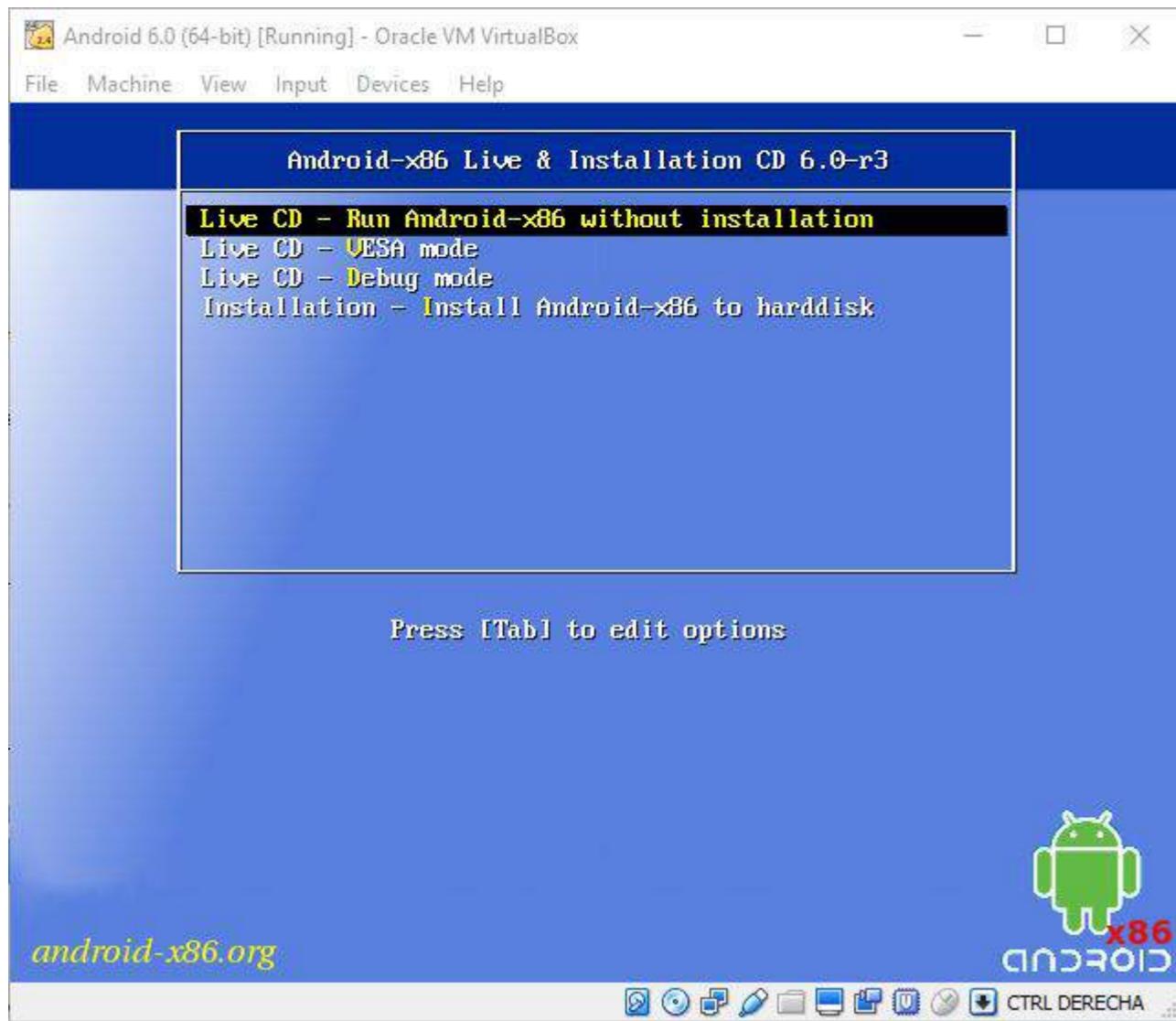
It doesn't cover VirtualBox or a Linux in detail but it shows the commands I've used to make it work.

Section 210.1: Virtual hard drive Setup for SDCARD Support

With the virtual hard drive just created, boot the virtual machine with the android-x86 image in the optical drive.



Once you boot, you can see the grub menu of the Live CD



选择调试模式选项，然后您应该会看到shell提示符。这是一个busybox shell。您可以通过切换虚拟控制台Alt-F1/F2/F3来获得更多shell。

使用fdisk创建两个分区（某些其他版本会使用cfdisk）。将它们格式化为ext3。然后重启：

```
# fdisk /dev/sda
```

然后输入：

"n" (新建分区)

"p" (主分区)

"1" (第1个分区)

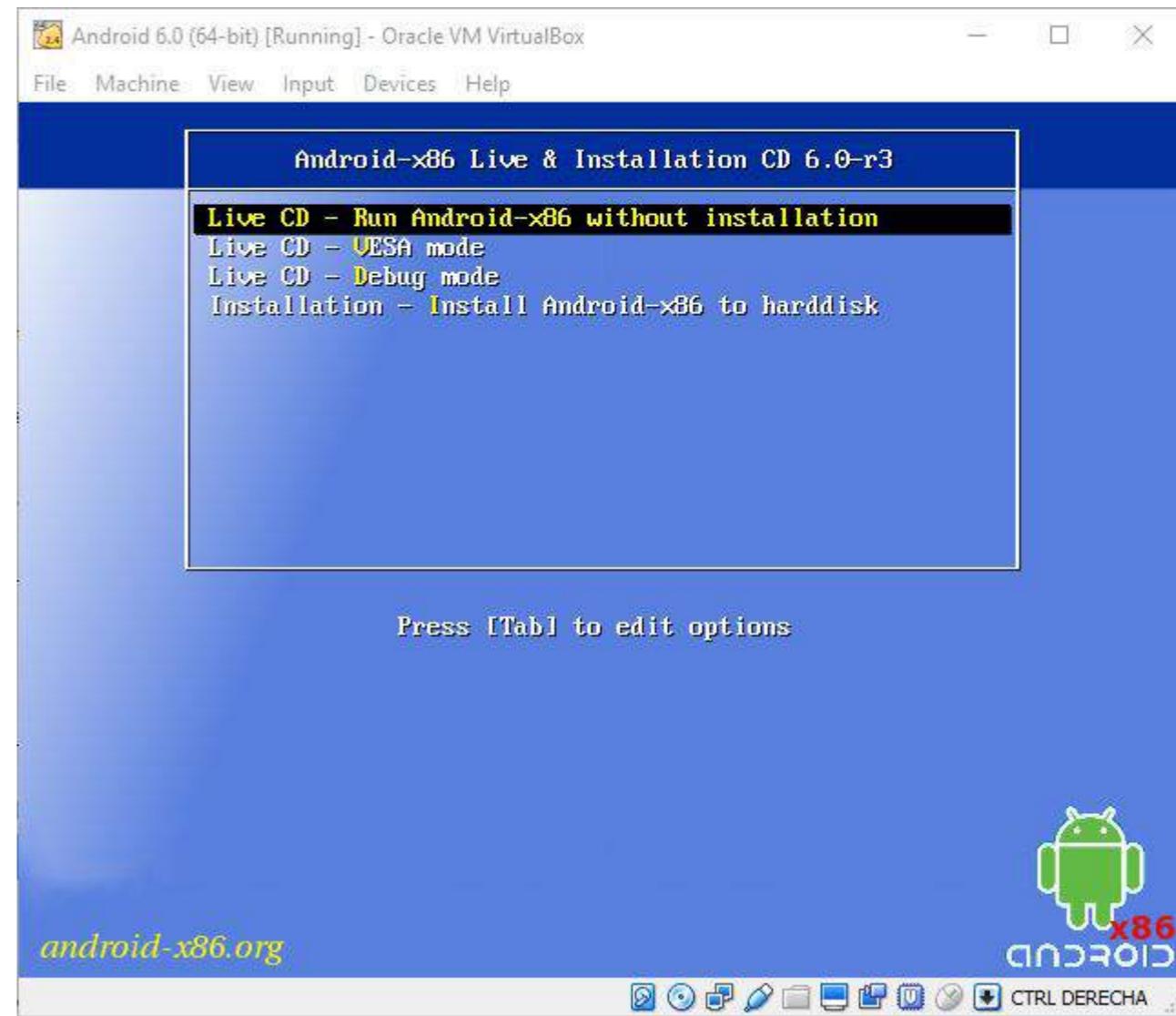
"1" (第一个柱面)

"261" (选择一个柱面，我们将为第二个分区保留50%的磁盘空间)

"2" (第二个分区)

"262" (第262个柱面)

"522" (选择最后一个柱面)



Choose the Debug Mode Option, then you should see the shell prompt. This is a busybox shell. You can get more shell by switching between virtual console Alt-F1/F2/F3.

Create two partitions by fdisk (some other versions would use cfdisk). Format them to ext3. Then reboot:

```
# fdisk /dev/sda
```

Then type:

"n" (new partition)

"p" (primary partition)

"1" (1st partition)

"1" (first cylinder)

"261" (choose a cylinder, we'll leave 50% of the disk for a 2nd partition)

"2" (2nd partition)

"262" (262nd cylinder)

"522" (choose the last cylinder)

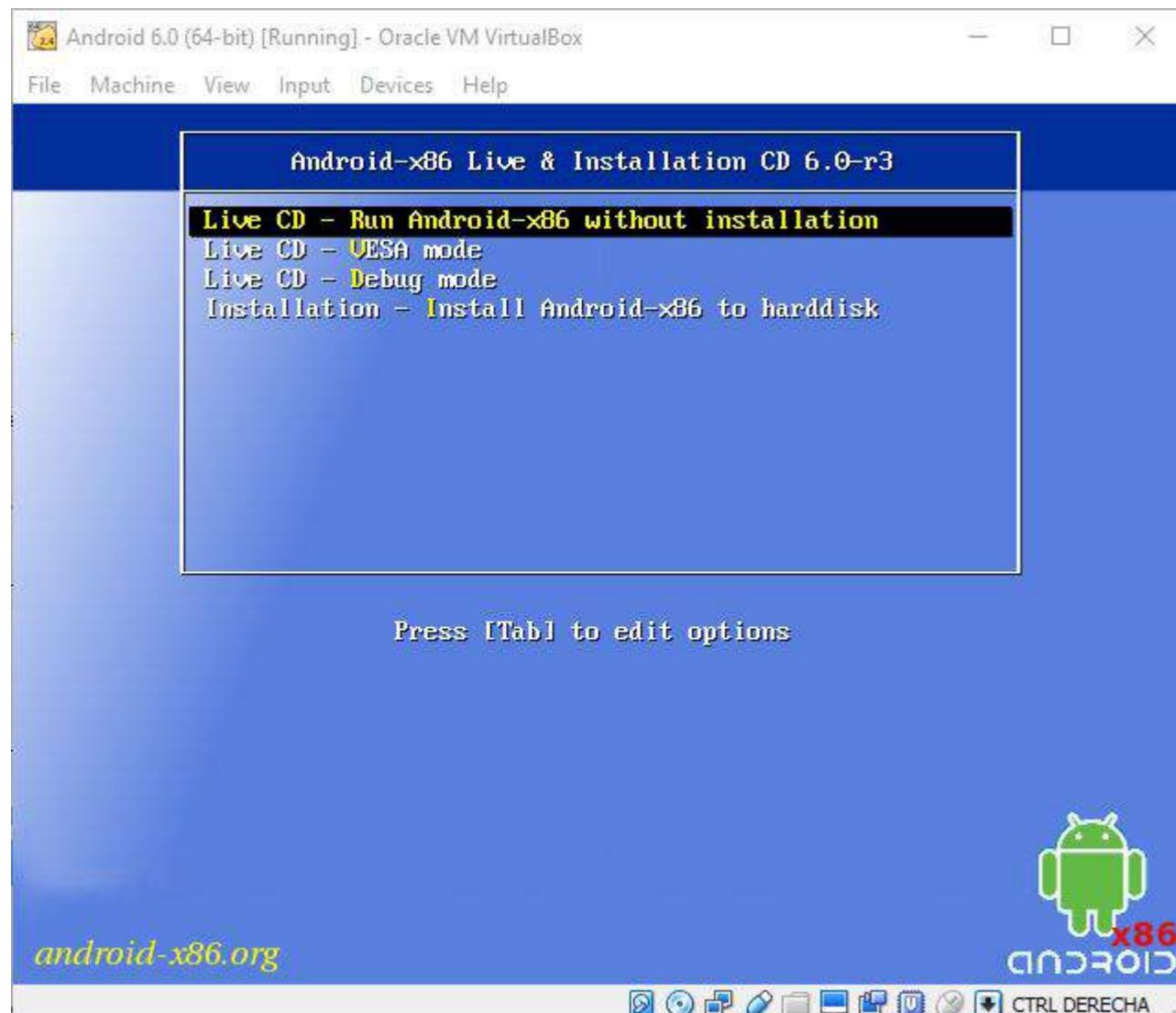
"w" (写入分区)

```
#mdev -s  
#mke2fs -j -L DATA /dev/sda1  
#mke2fs -j -L SDCARD /dev/sda2  
#reboot -f
```

当你重启虚拟机并出现grub菜单时，你可以编辑内核启动行，这样你就可以添加DATA=sda1 SDCARD=sda2选项来指向SD卡或数据分区。

第210.2节：分区安装

使用刚创建的虚拟硬盘，以android-x86镜像作为光驱启动虚拟机。



在Live CD的启动选项中选择“安装 - 将Android安装到硬盘”

选择sda1分区并安装Android，我们将安装grub。

重启虚拟机，但请确保镜像不在光驱中，以便它可以从虚拟硬盘启动。

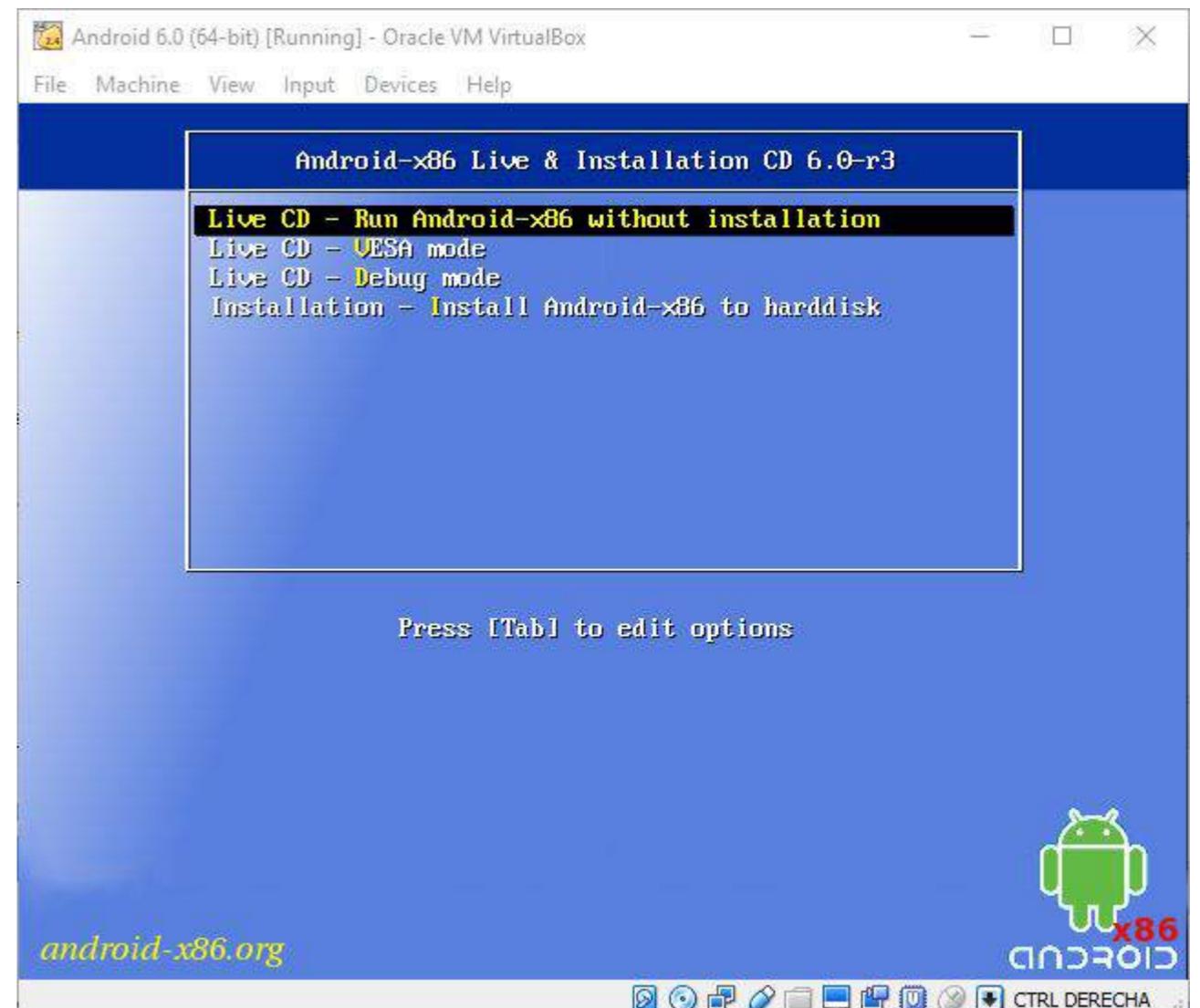
"w" (write the partition)

```
#mdev -s  
#mke2fs -j -L DATA /dev/sda1  
#mke2fs -j -L SDCARD /dev/sda2  
#reboot -f
```

When you restart the virtual machine and the grub menu appears and you will be able edit the kernel boot line so you can add DATA=sda1 SDCARD=sda2 options to point to the sdcard or the data partition.

Section 210.2: Installation in partition

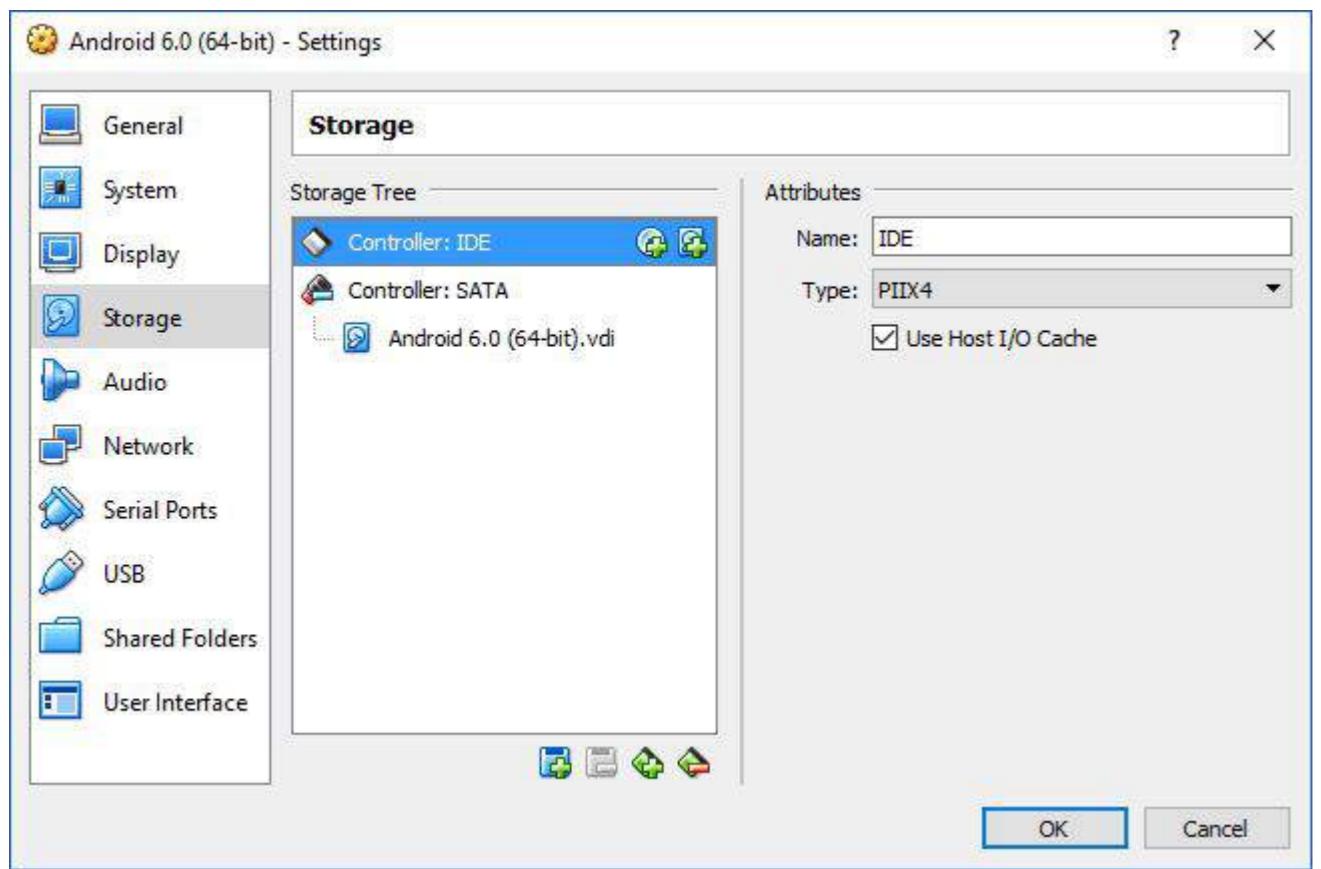
With the virtual hard drive just created, boot the virtual machine with the android-x86 image as the optical drive.



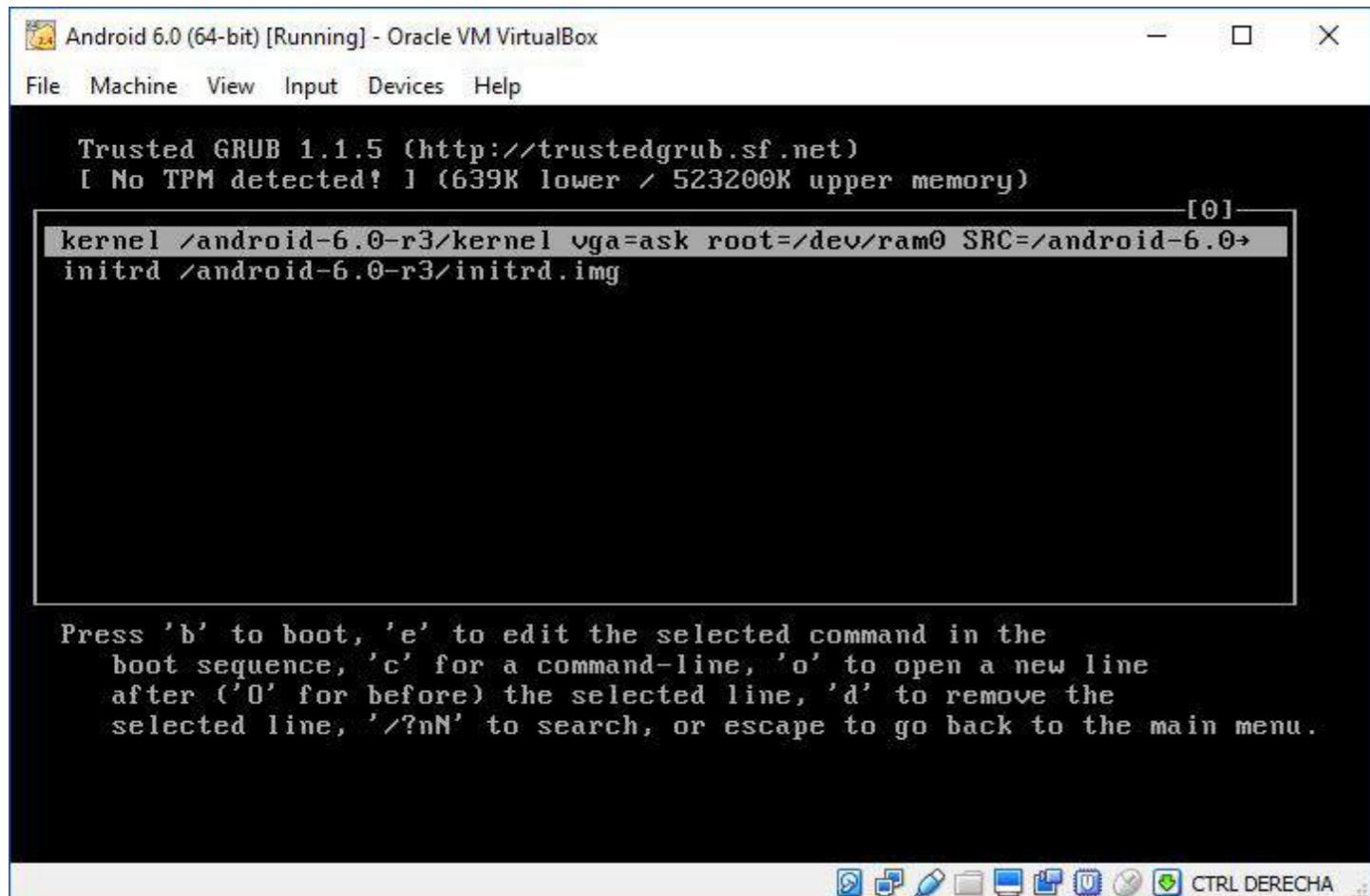
In the booting options of the Live CD choose "Installation - Install Android to hard disk"

Choose the sda1 partition and install android and we'll install grub.

Reboot the virtual machine but make sure that the image is not in the optical drive so it can restart from the virtual hard drive.

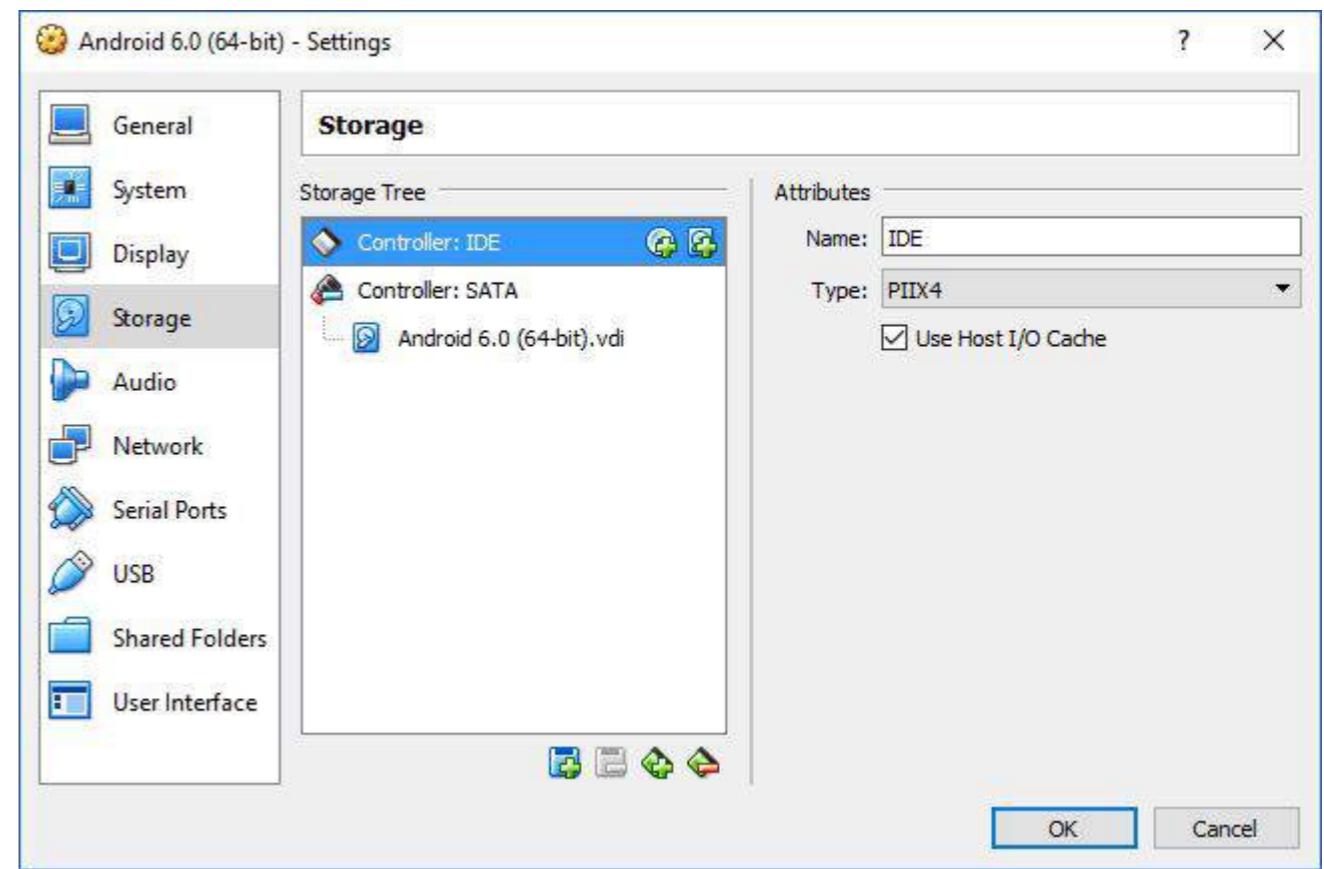


在grub菜单中，我们需要像选择“Android-x86 6.0-r3”选项那样编辑内核，所以按e键。

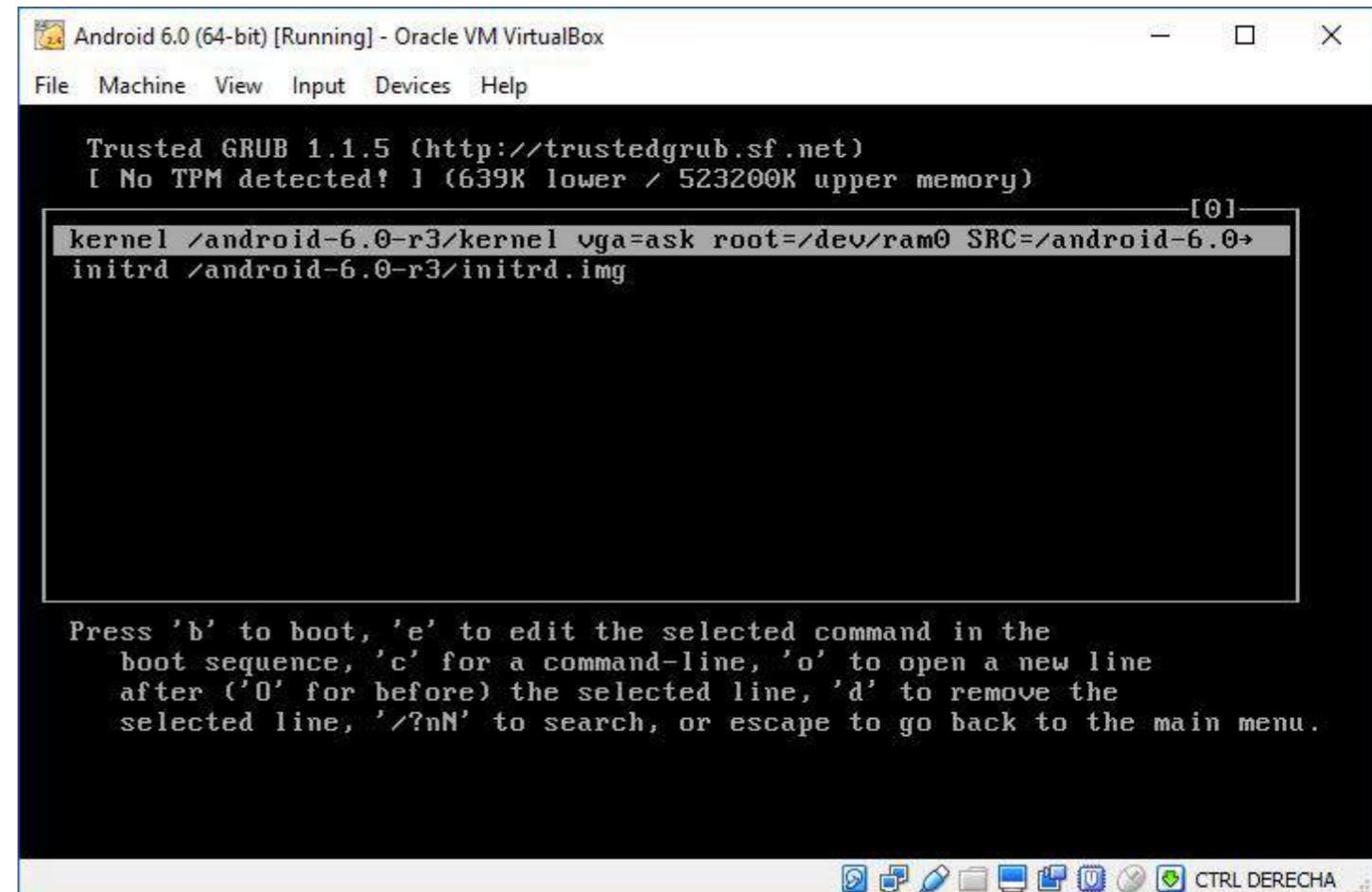


然后将“quiet”替换为“vga=ask”，并添加选项“SDCARD=sda2”

在我的情况下，修改后的内核行如下所示：



In the grub menu we need to edit kernel like in the "Android-x86 6.0-r3" option so press e.

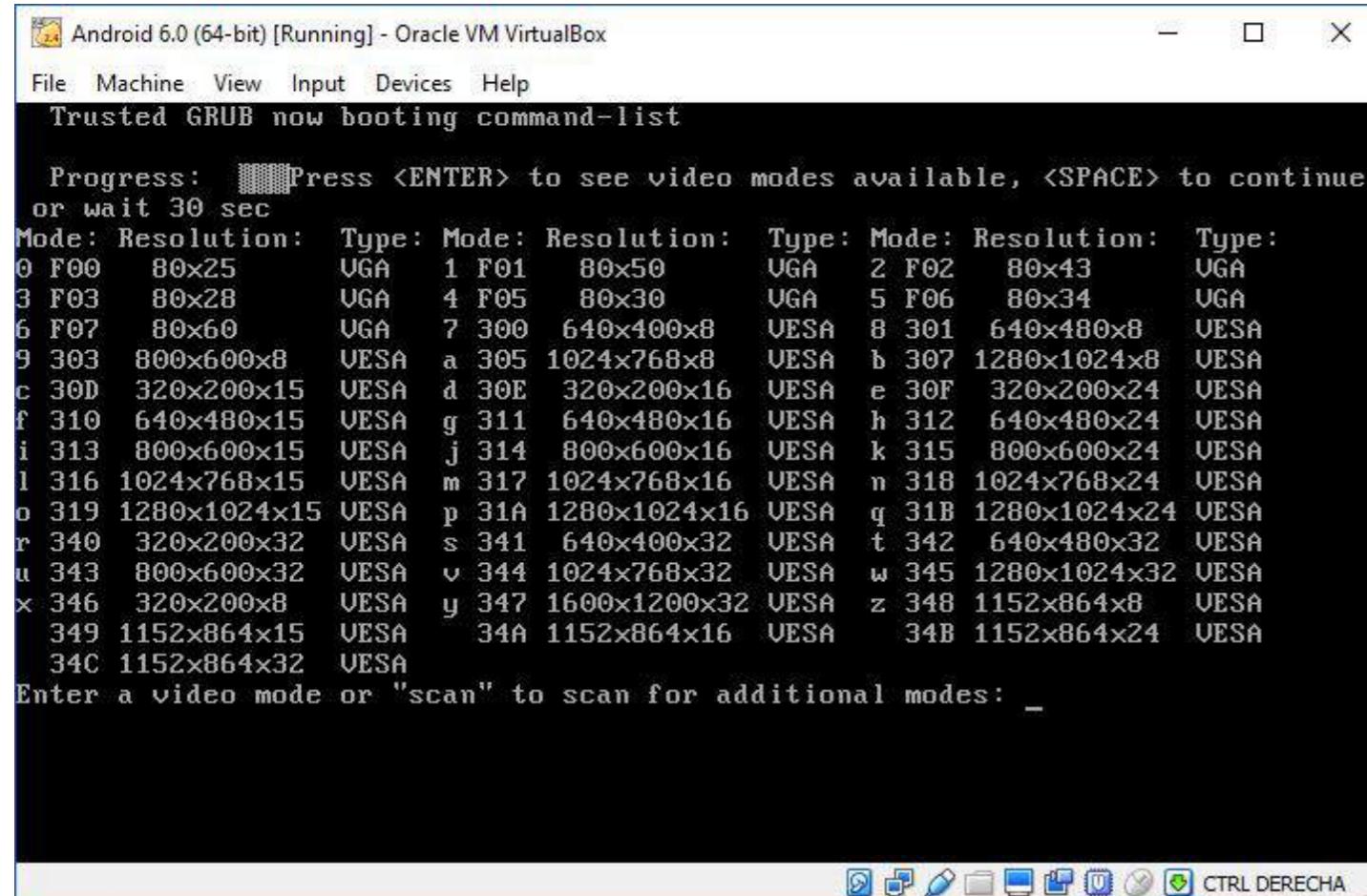


Then we substitute "quiet" with "vga=ask" and add the option "SDCARD=sda2"

In my case, the kernel line looks like this after modified:

```
kernel /android-6.0-r3/kernel vga=ask root=ram0 SRC=/android-6/android-6.0-r3 SDCARD=sda2
```

按b键启动，然后你可以通过按ENTER键选择屏幕大小（即vga=ask选项）



安装向导启动后选择语言。我可以选择英语（美国）和西班牙语（美国），选择其他语言时遇到了问题。

第210.3节：虚拟机设置

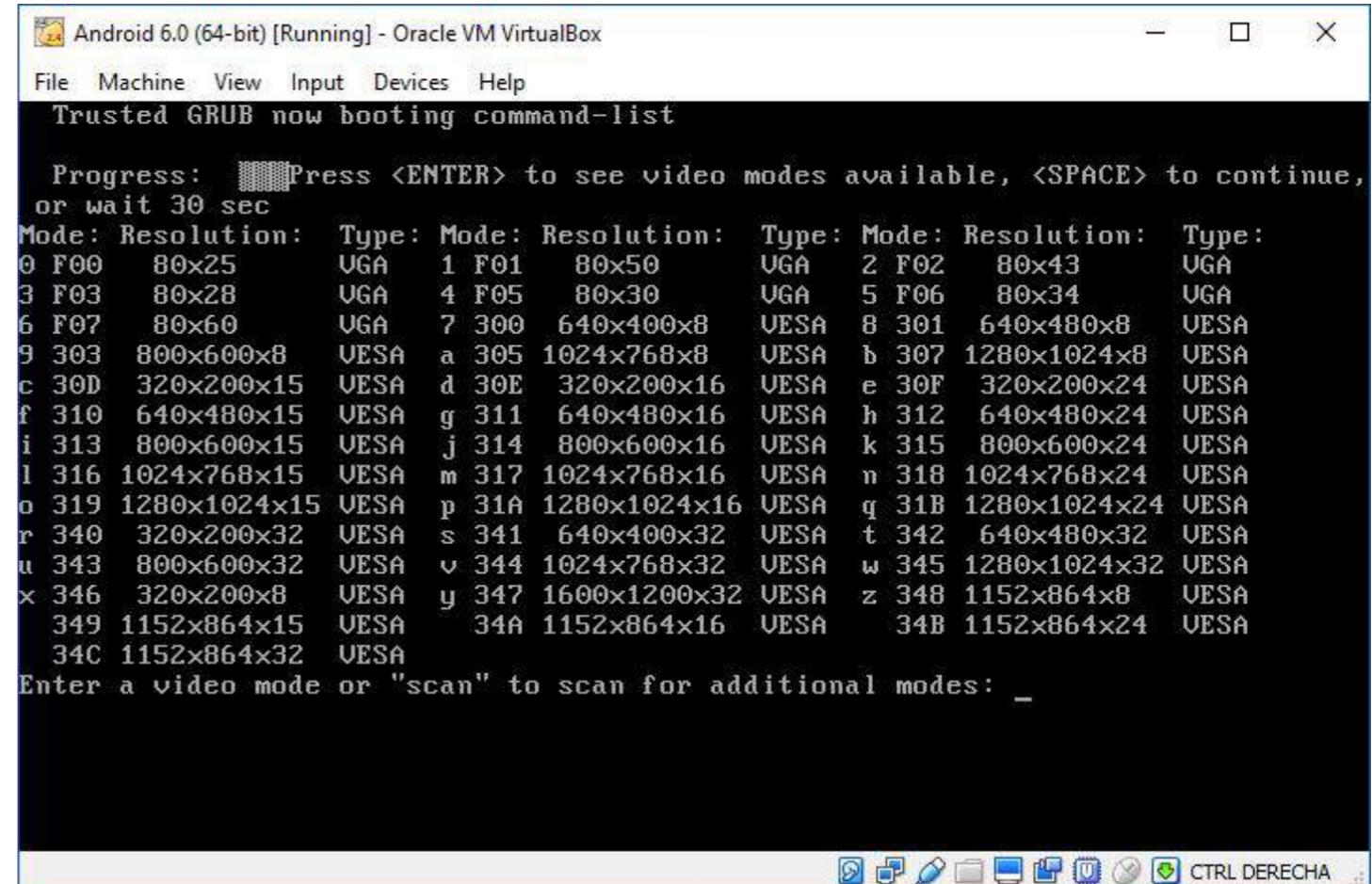
这是我的VirtualBox设置：

- 操作系统类型：Linux 2.6（我使用了64位，因为我的电脑支持）
- 虚拟硬盘大小：4Gb
- 内存：2048
- 显存：8M
- 声卡设备：Sound Blaster 16。
- 网络设备：PCnet-Fast III，连接到NAT。你也可以使用桥接适配器，但你的环境中需要有DHCP服务器。

此配置使用的镜像是android-x86_64-6.0-r3.iso（64位），下载自<http://www.android-x86.org/download>。我猜32位版本也能使用。

```
kernel /android-6.0-r3/kernel vga=ask root=ram0 SRC=/android-6/android-6.0-r3 SDCARD=sda2
```

Press b to boot, then you'll be able to choose the screen size pressing ENTER (the vga=ask option)



Once the installation wizard has started choose the language. I could choose English (United States) and Spanish (United States) and I had trouble choosing any other.

Section 210.3: Virtual Machine setup

These are my VirtualBox settings:

- OS Type: Linux 2.6 (I've user 64bit because my computer can support it)
- Virtual hard drive size: 4Gb
- Ram Memory: 2048
- Video Memory: 8M
- Sound device: Sound Blaster 16.
- Network device: PCnet-Fast III, attached to NAT. You can also use bridged adapter, but you need a DHCP server in your environment.

The image used with this configuration has been android-x86_64-6.0-r3.iso (it is 64bit) downloaded from <http://www.android-x86.org/download>. I suppose that it also works with 32bit version.

第211章：Leakcanary

Leak Canary是一个用于检测应用程序内存泄漏的Android和Java库

第211.1节：在Android

应用中实现Leak Canary

在你的build.gradle中需要添加以下依赖：

```
debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'  
releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'  
testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
```

在你的Application类中，需要在onCreate()方法内添加以下代码：

```
LeakCanary.install(this);
```

这就是你需要为LeakCanary做的全部工作，当你的构建中存在内存泄漏时，它会自动显示通知。

Chapter 211: Leakcanary

Leak Canary is an Android and Java library used to detect leak in the application

Section 211.1: Implementing a Leak Canary in Android Application

In your *build.gradle* you need to add the below dependencies:

```
debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'  
releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'  
testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
```

In your Application class you need to add the below code inside your *onCreate()*:

```
LeakCanary.install(this);
```

That's all you need to do for *LeakCanary*, it will automatically show notifications when there is a leak in your build.

第212章：Okio

第212.1节：下载 / 实现

下载最新的JAR包或通过Maven获取：

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

或者使用Gradle：

```
compile 'com.squareup.okio:okio:1.12.0'
```

第212.2节：PNG解码器

解码PNG文件的块展示了Okio的实际应用。

```
private static final ByteString PNG_HEADER = ByteString.decodeHex("89504e470d0a1a0a");

public void decodePng(InputStream in) throws IOException {
    try (BufferedSource pngSource = Okio.buffer(Okio.source(in))) {
        ByteString header = pngSource.readByteString(PNG_HEADER.size());
        if (!header.equals(PNG_HEADER)) {
            throw new IOException("Not a PNG.");
        }

        while (true) {
            Buffer chunk = new Buffer();

            // 每个块包含长度、类型、数据和CRC偏移。
            int length = pngSource.readInt();
            String type = pngSource.readUtf8(4);
            pngSource.readFully(chunk, length);
            int crc = pngSource.readInt();

            decodeChunk(类型, 块);
            if (type.equals("IEND")) break;
        }
    }
}

private void decodeChunk(String type, Buffer chunk) {
    if (type.equals("IHDR")) {
        int 宽度 = chunk.readInt();
        int 高度 = chunk.readInt();
        System.out.printf("%08x: %s %d x %d%n", chunk.size(), type, 宽度, 高度);
    } else {
        System.out.printf("%08x: %s%n", chunk.size(), type);
    }
}
```

第212.3节：字节串和缓冲区

字节串和缓冲区

Chapter 212: Okio

Section 212.1: Download / Implement

Download the latest JAR or grab via Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

or Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

Section 212.2: PNG decoder

Decoding the chunks of a PNG file demonstrates Okio in practice.

```
private static final ByteString PNG_HEADER = ByteString.decodeHex("89504e470d0a1a0a");

public void decodePng(InputStream in) throws IOException {
    try (BufferedSource pngSource = Okio.buffer(Okio.source(in))) {
        ByteString header = pngSource.readByteString(PNG_HEADER.size());
        if (!header.equals(PNG_HEADER)) {
            throw new IOException("Not a PNG.");
        }

        while (true) {
            Buffer chunk = new Buffer();

            // Each chunk is a length, type, data, and CRC offset.
            int length = pngSource.readInt();
            String type = pngSource.readUtf8(4);
            pngSource.readFully(chunk, length);
            int crc = pngSource.readInt();

            decodeChunk(type, chunk);
            if (type.equals("IEND")) break;
        }
    }
}

private void decodeChunk(String type, Buffer chunk) {
    if (type.equals("IHDR")) {
        int width = chunk.readInt();
        int height = chunk.readInt();
        System.out.printf("%08x: %s %d x %d%n", chunk.size(), type, width, height);
    } else {
        System.out.printf("%08x: %s%n", chunk.size(), type);
    }
}
```

Section 212.3: ByteStrings and Buffers

ByteStrings and Buffers

Okio围绕两种类型构建，这两种类型通过简单的API提供了丰富的功能：

ByteString 是一个不可变的字节序列。对于字符数据，String 是基础。ByteString 是 String 失散已久的兄弟，使得将二进制数据作为值来处理变得简单。这个类使用方便：它知道如何将自身编码和解码为十六进制、base64 和 UTF-8。

Buffer 是一个可变的字节序列。像 ArrayList 一样，你不需要提前确定缓冲区大小。你可以将缓冲区作为队列来读写：将数据写入末尾，从前端读取。无需管理位置、限制或容量。

在内部，ByteString 和 Buffer 做了一些巧妙的处理以节省 CPU 和内存。如果你将 UTF-8 字符串编码为 ByteString，它会缓存对该字符串的引用，这样如果稍后解码，就无需额外操作。

Buffer 实现为一个分段的链表。当你将数据从一个缓冲区移动到另一个缓冲区时，它会重新分配分段的所有权，而不是复制数据。这种方法对多线程程序特别有用：与网络通信的线程可以与工作线程交换数据，无需任何复制或复杂操作。

Okio is built around two types that pack a lot of capability into a straightforward API:

ByteString is an immutable sequence of bytes. For character data, String is fundamental. ByteString is String's long-lost brother, making it easy to treat binary data as a value. This class is ergonomic: it knows how to encode and decode itself as hex, base64, and UTF-8.

Buffer is a mutable sequence of bytes. Like ArrayList, you don't need to size your buffer in advance. You read and write buffers as a queue: write data to the end and read it from the front. There's no obligation to manage positions, limits, or capacities.

Internally, ByteString and Buffer do some clever things to save CPU and memory. If you encode a UTF-8 string as a ByteString, it caches a reference to that string so that if you decode it later, there's no work to do.

Buffer is implemented as a linked list of segments. When you move data from one buffer to another, it reassigns ownership of the segments rather than copying the data across. This approach is particularly helpful for multithreaded programs: a thread that talks to the network can exchange data with a worker thread without any copying or ceremony.

第213章：蓝牙低功耗

本文件旨在作为对原始文档的补充，重点介绍 Android 5.0 (API 21) 引入的最新蓝牙低功耗 (BLE) API。内容涵盖中央和外围角色，以及如何开始扫描和广播操作。

第213.1节：查找 BLE 设备

使用蓝牙 API 需要以下权限：

```
android.permission.BLUETOOTH android.permission.BLUETOOTH_ADMIN
```

如果你的目标设备是 Android 6.0 (API 级别 23) 或更高版本，并且想执行扫描/广播操作，则需要位置权限：

```
android.permission.ACCESS_FINE_LOCATION
```

或

```
android.permission.ACCESS_COARSE_LOCATION
```

注意：运行 Android 6.0 (API 级别 23) 或更高版本的设备还需要启用定位服务。

启动扫描/广播操作需要一个 BluetoothAdapter 对象：

```
BluetoothManager bluetoothManager = (BluetoothManager)  
context.getSystemService(Context.BLUETOOTH_SERVICE);  
bluetoothAdapter = bluetoothManager.getAdapter();
```

BluetoothLeScanner 类的 startScan (ScanCallback callback)方法是启动扫描操作的最基本方式。需要一个 ScanCallback 对象来接收结果：

```
bluetoothAdapter.getBluetoothLeScanner().startScan(new ScanCallback() {  
    @Override  
    public void onScanResult(int callbackType, ScanResult result) {  
        super.onScanResult(callbackType, result);  
        Log.i(TAG, "远程设备名称: " + result.getDevice().getName());  
    }  
});
```

第 213.2 节：连接到 GATT 服务器

一旦发现了所需的 BluetoothDevice 对象，可以使用其 connectGatt()方法进行连接，该方法的参数包括一个 Context 对象、一个表示是否自动连接到BLE 设备的布尔值，以及一个 BluetoothGattCallback 引用，用于接收连接事件和客户端操作结果：

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {  
    device.connectGatt(context, false, bluetoothGattCallback, BluetoothDevice.TRANSPORT_AUTO);  
} else {  
    device.connectGatt(context, false, bluetoothGattCallback);  
}
```

重写 BluetoothGattCallback 中的 onConnectionStateChange 以接收连接和断开连接事件：

Chapter 213: Bluetooth Low Energy

This documentation is meant as an enhancement over the [original documentation](#) and it will focus on the latest Bluetooth LE API introduced in Android 5.0 (API 21). Both Central and Peripheral roles will be covered as well as how to start scanning and advertising operations.

Section 213.1: Finding BLE Devices

The following permissions are required to use the Bluetooth APIs:

```
android.permission.BLUETOOTH android.permission.BLUETOOTH_ADMIN
```

If you're targeting devices with Android 6.0 (**API Level 23**) or higher and want to perform scanning/advertising operations you will require a Location permission:

```
android.permission.ACCESS_FINE_LOCATION
```

or

```
android.permission.ACCESS_COARSE_LOCATION
```

Note:- Devices with Android 6.0 (API Level 23) or higher also need to have Location Services enabled.

A BluetoothAdapter object is required to start scanning/advertising operations:

```
BluetoothManager bluetoothManager = (BluetoothManager)  
context.getSystemService(Context.BLUETOOTH_SERVICE);  
bluetoothAdapter = bluetoothManager.getAdapter();
```

The startScan (ScanCallback callback)method of the BluetoothLeScanner class is the most basic way to start a scanning operation. A ScanCallback object is required to receive results:

```
bluetoothAdapter.getBluetoothLeScanner().startScan(new ScanCallback() {  
    @Override  
    public void onScanResult(int callbackType, ScanResult result) {  
        super.onScanResult(callbackType, result);  
        Log.i(TAG, "Remote device name: " + result.getDevice().getName());  
    }  
});
```

Section 213.2: Connecting to a GATT Server

Once you have discovered a desired BluetoothDevice object, you can connect to it by using its connectGatt() method which takes as parameters a Context object, a boolean indicating whether to automatically connect to the BLE device and a BluetoothGattCallback reference where connection events and client operations results will be delivered:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {  
    device.connectGatt(context, false, bluetoothGattCallback, BluetoothDevice.TRANSPORT_AUTO);  
} else {  
    device.connectGatt(context, false, bluetoothGattCallback);  
}
```

Override onConnectionStateChange in BluetoothGattCallback to receive connection an disconnection events:

```

BluetoothGattCallback bluetoothGattCallback =
    new BluetoothGattCallback() {
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "已连接到 GATT 服务器。");
    } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {

        Log.i(TAG, "已从 GATT 服务器断开连接。");
    }
}
};

```

第213.3节：从特征写入和读取

一旦连接到Gatt服务器，您将通过写入和读取服务器的特征来与其交互。为此，首先必须发现该服务器上有哪些服务以及每个服务中有哪些特征：

```

@Override
public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "已连接到GATT服务器。");
        gatt.discoverServices();
    }
}

@Override
public void onServicesDiscovered(BluetoothGatt gatt, int status) {
    if (status == BluetoothGatt.GATT_SUCCESS) {
        List<BluetoothGattService> services = gatt.getServices();
        for (BluetoothGattService service : services) {
            List<BluetoothGattCharacteristic> characteristics =
service.getCharacteristics();
            for (BluetoothGattCharacteristic characteristic : characteristics) {
                //一旦获得特征对象，您就可以对其进行读/写
                //操作
            }
        }
    }
}

```

一个基本的写操作如下：

```

characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);

```

当写入过程完成后，onCharacteristicWrite 方法将在你的 BluetoothGattCallback 中被调用：

```

@Override
public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
    super.onCharacteristicWrite(gatt, characteristic, status);
}

```

```

BluetoothGattCallback bluetoothGattCallback =
    new BluetoothGattCallback() {
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "Connected to GATT server.");
    } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
        Log.i(TAG, "Disconnected from GATT server.");
    }
}
};

```

Section 213.3: Writing and Reading from Characteristics

Once you are connected to a Gatt Server, you're going to be interacting with it by writing and reading from the server's characteristics. To do this, first you have to discover what services are available on this server and which characteristics are available in each service:

```

@Override
public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "Connected to GATT server.");
        gatt.discoverServices();
    }
}

@Override
public void onServicesDiscovered(BluetoothGatt gatt, int status) {
    if (status == BluetoothGatt.GATT_SUCCESS) {
        List<BluetoothGattService> services = gatt.getServices();
        for (BluetoothGattService service : services) {
            List<BluetoothGattCharacteristic> characteristics =
service.getCharacteristics();
            for (BluetoothGattCharacteristic characteristic : characteristics) {
                //Once you have a characteristic object, you can perform read/write
                //operations with it
            }
        }
    }
}

```

A basic write operation goes like this:

```

characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);

```

When the write process has finished, the onCharacteristicWrite method of your BluetoothGattCallback will be called:

```

@Override
public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
    super.onCharacteristicWrite(gatt, characteristic, status);
}

```

```
Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);  
}
```

一个基本的写操作如下：

```
gatt.readCharacteristic(characteristic);
```

当写入过程完成后，onCharacteristicRead 方法将在你的 BluetoothGattCallback 中被调用：

```
@Override  
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic,  
int status) {  
    super.onCharacteristicRead(gatt, characteristic, status);  
    byte[] value = characteristic.getValue();  
}
```

第213.4节：订阅来自Gatt服务器的通知

当特征值发生变化时，您可以请求从Gatt服务器接收通知：

```
gatt.setCharacteristicNotification(characteristic, true);  
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(  
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb"));  
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);  
mBluetoothGatt.writeDescriptor(descriptor);
```

所有来自服务器的通知都会在您的

BluetoothGattCallback的onCharacteristicChanged方法中接收：

```
@Override  
public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic  
characteristic) {  
    super.onCharacteristicChanged(gatt, characteristic);  
    byte[] newValue = characteristic.getValue();  
}
```

第213.5节：广播BLE设备

您可以使用蓝牙低功耗广告（Bluetooth LE Advertising）向所有附近设备广播数据包，而无需先建立连接。请注意，广告数据严格限制为31字节。广播您的设备也是让其他用户连接您的第一步。

由于并非所有设备都支持蓝牙低功耗广告，第一步是检查您的设备是否具备支持该功能的所有必要条件。之后，您可以初始化一个BluetoothLeAdvertiser对象，并通过它开始广告操作：

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP &&  
bluetoothAdapter.isMultipleAdvertisementSupported())  
{  
    BluetoothLeAdvertiser advertiser = bluetoothAdapter.getBluetoothLeAdvertiser();  
  
    AdvertiseData.Builder dataBuilder = new AdvertiseData.Builder();  
    //根据您的需求定义服务UUID  
    dataBuilder.addServiceUuid(SERVICE_UUID);
```

```
Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);  
}
```

A basic write operation goes like this:

```
gatt.readCharacteristic(characteristic);
```

When the write process has finished, the onCharacteristicRead method of your BluetoothGattCallback will be called:

```
@Override  
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic,  
int status) {  
    super.onCharacteristicRead(gatt, characteristic, status);  
    byte[] value = characteristic.getValue();  
}
```

Section 213.4: Subscribing to Notifications from the Gatt Server

You can request to be notified from the Gatt Server when the value of a characteristic has been changed:

```
gatt.setCharacteristicNotification(characteristic, true);  
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(  
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb"));  
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);  
mBluetoothGatt.writeDescriptor(descriptor);
```

All notifications from the server will be received in the onCharacteristicChanged method of your BluetoothGattCallback:

```
@Override  
public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic  
characteristic) {  
    super.onCharacteristicChanged(gatt, characteristic);  
    byte[] newValue = characteristic.getValue();  
}
```

Section 213.5: Advertising a BLE Device

You can use Bluetooth LE Advertising to broadcast data packages to all nearby devices without having to establish a connection first. Bear in mind that there's a strict limit of 31 bytes of advertisement data. Advertising your device is also the first step towards letting other users connect to you.

Since not all devices support Bluetooth LE Advertising, the first step is to check that your device has all the necessary requirements to support it. Afterwards, you can initialize a BluetoothLeAdvertiser object and with it, you can start advertising operations:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP &&  
bluetoothAdapter.isMultipleAdvertisementSupported())  
{  
    BluetoothLeAdvertiser advertiser = bluetoothAdapter.getBluetoothLeAdvertiser();  
  
    AdvertiseData.Builder dataBuilder = new AdvertiseData.Builder();  
    //Define a service UUID according to your needs  
    dataBuilder.addServiceUuid(SERVICE_UUID);
```

```

dataBuilder.setIncludeDeviceName(true);

AdvertiseSettings.Builder settingsBuilder = new AdvertiseSettings.Builder();
    settingsBuilder.setAdvertiseMode(AdvertiseSettings.ADVERTISE_MODE_LOW_POWER);
    settingsBuilder.setTimeout(0);

    //如果您打算开启Gatt服务器以允许远程连接到您的设备,请使用可连接标志。

settingsBuilder.setConnectable(true);

    AdvertiseCallback advertiseCallback=new AdvertiseCallback() {
        @Override
        public void onStartSuccess(AdvertiseSettings settingsInEffect) {
            super.onStartSuccess(settingsInEffect);
            Log.i(TAG, "onStartSuccess: ");
        }

        @Override
        public void onStartFailure(int errorCode) {
            super.onStartFailure(errorCode);
            Log.e(TAG, "onStartFailure: "+errorCode );
        }
    };
advertising.startAdvertising(settingsBuilder.build(),dataBuilder.build(),advertiseCallback);
}

```

第213.6节：使用Gatt服务器

为了让您的设备作为外围设备，首先需要打开一个BluetoothGattServer，并至少添加一个BluetoothGattService和一个BluetoothGattCharacteristic：

```

BluetoothGattServer server=bluetoothManager.openGattServer(context, bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
    BluetoothGattService.SERVICE_TYPE_PRIMARY);

BluetoothGattCharacteristic characteristic = new BluetoothGattCharacteristic(CHARACTERISTIC_UUID,
    BluetoothGattCharacteristic.PROPERTY_READ |
    BluetoothGattCharacteristic.PROPERTY_WRITE |
    BluetoothGattCharacteristic.PROPERTY_NOTIFY,
    BluetoothGattCharacteristic.PERMISSION_READ |
    BluetoothGattCharacteristic.PERMISSION_WRITE);

characteristic.addDescriptor(new
    BluetoothGattDescriptor(UUID.fromString("00002902-0000-1000-8000-00805f9b34fb"),
    BluetoothGattCharacteristic.PERMISSION_WRITE));

service.addCharacteristic(characteristic);

server.addService(service);

```

BluetoothGattServerCallback负责接收所有与您的BluetoothGattServer相关的事件：

```

BluetoothGattServerCallback bluetoothGattServerCallback= new BluetoothGattServerCallback() {
    @Override

```

```

dataBuilder.setIncludeDeviceName(true);

AdvertiseSettings.Builder settingsBuilder = new AdvertiseSettings.Builder();
    settingsBuilder.setAdvertiseMode(AdvertiseSettings.ADVERTISE_MODE_LOW_POWER);
    settingsBuilder.setTimeout(0);

    //Use the connectable flag if you intend on opening a Gatt Server
    //to allow remote connections to your device.
    settingsBuilder.setConnectable(true);

    AdvertiseCallback advertiseCallback=new AdvertiseCallback() {
        @Override
        public void onStartSuccess(AdvertiseSettings settingsInEffect) {
            super.onStartSuccess(settingsInEffect);
            Log.i(TAG, "onStartSuccess: ");
        }

        @Override
        public void onStartFailure(int errorCode) {
            super.onStartFailure(errorCode);
            Log.e(TAG, "onStartFailure: "+errorCode );
        }
    };
advertising.startAdvertising(settingsBuilder.build(),dataBuilder.build(),advertiseCallback);
}

```

Section 213.6: Using a Gatt Server

In order for your device to act as a peripheral, first you need to open a BluetoothGattServer and populate it with at least one BluetoothGattService and one BluetoothGattCharacteristic:

```

BluetoothGattServer server=bluetoothManager.openGattServer(context, bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
    BluetoothGattService.SERVICE_TYPE_PRIMARY);

```

This is an example of a BluetoothGattCharacteristic with full write,read and notify permissions. According to your needs, you might want to fine tune the permissions that you grant this characteristic:

```

BluetoothGattCharacteristic characteristic = new BluetoothGattCharacteristic(CHARACTERISTIC_UUID,
    BluetoothGattCharacteristic.PROPERTY_READ |
    BluetoothGattCharacteristic.PROPERTY_WRITE |
    BluetoothGattCharacteristic.PROPERTY_NOTIFY,
    BluetoothGattCharacteristic.PERMISSION_READ |
    BluetoothGattCharacteristic.PERMISSION_WRITE);

characteristic.addDescriptor(new
    BluetoothGattDescriptor(UUID.fromString("00002902-0000-1000-8000-00805f9b34fb"),
    BluetoothGattCharacteristic.PERMISSION_WRITE));

service.addCharacteristic(characteristic);

server.addService(service);

```

The BluetoothGattServerCallback is responsible for receiving all events related to your BluetoothGattServer:

```

BluetoothGattServerCallback bluetoothGattServerCallback= new BluetoothGattServerCallback() {
    @Override

```

```

public void onConnectionStateChange(BluetoothDevice device, int status, int
newState) {
    super.onConnectionStateChange(device, status, newState);
}

@Override
public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
}

@Override
public void onCharacteristicWriteRequest(BluetoothDevice 设备, int requestId,
BluetoothGattCharacteristic 特征, boolean preparedWrite, boolean responseNeeded, int
offset, byte[] value) {
    super.onCharacteristicWriteRequest(device, requestId, characteristic,
preparedWrite, responseNeeded, offset, value);
}

@Override
public void onDescriptorReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattDescriptor descriptor) {
    super.onDescriptorReadRequest(device, requestId, offset, descriptor);
}

@Override
public void onDescriptorWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattDescriptor descriptor, boolean preparedWrite, boolean responseNeeded, int offset,
byte[] value) {
    super.onDescriptorWriteRequest(device, requestId, descriptor, preparedWrite,
responseNeeded, offset, value);
}

```

每当你收到对特征或描述符的写入/读取请求时，必须发送响应以使请求成功完成：

```

@Override
public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int offset,
BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
    server.sendResponse(device, requestId, BluetoothGatt.GATT_SUCCESS, offset, YOUR_RESPONSE);
}

```

```

public void onConnectionStateChange(BluetoothDevice device, int status, int
newState) {
    super.onConnectionStateChange(device, status, newState);
}

@Override
public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
}

@Override
public void onCharacteristicWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattCharacteristic characteristic, boolean preparedWrite, boolean responseNeeded, int
offset, byte[] value) {
    super.onCharacteristicWriteRequest(device, requestId, characteristic,
preparedWrite, responseNeeded, offset, value);
}

@Override
public void onDescriptorReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattDescriptor descriptor) {
    super.onDescriptorReadRequest(device, requestId, offset, descriptor);
}

@Override
public void onDescriptorWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattDescriptor descriptor, boolean preparedWrite, boolean responseNeeded, int offset,
byte[] value) {
    super.onDescriptorWriteRequest(device, requestId, descriptor, preparedWrite,
responseNeeded, offset, value);
}

```

Whenever you receive a request for a write/read to a characteristic or descriptor you must send a response to it in order for the request to be completed successfully:

```

@Override
public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int offset,
BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
    server.sendResponse(device, requestId, BluetoothGatt.GATT_SUCCESS, offset, YOUR_RESPONSE);
}

```

第214章：Looper

Looper 是一个 Android 类，用于为线程运行消息循环，通常线程本身没有关联的消息循环。

Android 中最常见的 Looper 是主循环，也常被称为主线程。该实例对于一个应用程序是唯一的，可以通过 `Looper.getMainLooper()` 静态访问。

如果当前线程关联了 Looper，可以通过 `Looper.myLooper()` 获取。

第214.1节：创建一个简单的 LooperThread

官方文档给出的 Looper 线程实现的典型示例使用了 `Looper.prepare()` 和 `Looper.loop()`，并在这两个调用之间关联了一个 Handler。

```
class LooperThread extends Thread {  
    public Handler mHandler;  
  
    public void run() {  
        Looper.prepare();  
  
        mHandler = new Handler() {  
            public void handleMessage(Message msg) {  
                // 在这里处理接收到的消息  
            }  
        };  
  
        Looper.loop();  
    }  
}
```

第214.2节：使用HandlerThread运行循环

A `HandlerThread` 可以用来启动一个带有 Looper 的线程。这个 Looper 随后可以用来创建一个 Handler，以便与其通信。

```
HandlerThread thread = new HandlerThread("thread-name");  
thread.start();  
Handler handler = new Handler(thread.getLooper());
```

Chapter 214: Looper

A `Looper` is an Android class used to run a message loop for a thread, which usually do not have one associated with them.

The most common Looper in Android is the main-loop, also commonly known as the main-thread. This instance is unique for an application and can be accessed statically with `Looper.getMainLooper()`.

If a Looper is associated with the current thread, it can be retrieved with `Looper.myLooper()`.

Section 214.1: Create a simple LooperThread

A typical example of the implementation of a Looper thread given by the official documentation uses `Looper.prepare()` and `Looper.loop()` and associates a Handler with the loop between these calls.

```
class LooperThread extends Thread {  
    public Handler mHandler;  
  
    public void run() {  
        Looper.prepare();  
  
        mHandler = new Handler() {  
            public void handleMessage(Message msg) {  
                // process incoming messages here  
            }  
        };  
  
        Looper.loop();  
    }  
}
```

Section 214.2: Run a loop with a HandlerThread

A `HandlerThread` can be used to start a thread with a Looper. This looper then can be used to create a Handler for communications with it.

```
HandlerThread thread = new HandlerThread("thread-name");  
thread.start();  
Handler handler = new Handler(thread.getLooper());
```

第215章：注解处理器

注解处理器是javac内置的一个工具，用于在编译时扫描和处理注解。

注解是一类元数据，可以关联到类、方法、字段，甚至其他注解。访问这些注解有两种方式：运行时通过反射，编译时通过注解处理器。

215.1节：@NonNull 注解

```
public class Foo {  
    private String name;  
    public Foo(@NonNull String name){...};  
    ...  
}
```

这里 @NonNull 是一个注解，由 Android Studio 在编译时处理，用来提醒你该函数需要非空参数。

第215.2节：注解的类型

注解有三种类型。

1. 标记注解 - 没有方法的注解

```
@interface CustomAnnotation {}
```

2. 单值注解 - 只有一个方法的注解

```
@interface CustomAnnotation {  
    int value();  
}
```

3. 多值注解 - 有多个方法的注解

```
@interface CustomAnnotation{  
    int value1();  
    String value2();  
    String value3();  
}
```

第215.3节：创建和使用自定义注解

创建自定义注解时，我们需要决定

- 目标 - 注解将作用于哪些层级，如字段级、方法级、类型级等。
- 保留策略 - 注解将保留到什么程度。

为此，我们内置了自定义注解。请查看这些常用的注解：

@Target

Chapter 215: Annotation Processor

Annotation processor is a tool build in javac for scanning and processing annotations at compile time.

Annotations are a class of metadata that can be associated with classes, methods, fields, and even other annotations. There are two ways to access these annotations at runtime via reflection and at compile time via annotation processors.

Section 215.1: @NonNull Annotation

```
public class Foo {  
    private String name;  
    public Foo(@NonNull String name){...};  
    ...  
}
```

Here @NonNull is annotation which is processed compile time by the android studio to warn you that the particular function needs non null parameter.

Section 215.2: Types of Annotations

There are three types of annotations.

1. **Marker Annotation** - annotation that has no method

```
@interface CustomAnnotation {}
```

2. **Single-Value Annotation** - annotation that has one method

```
@interface CustomAnnotation {  
    int value();  
}
```

3. **Multi-Value Annotation** - annotation that has more than one method

```
@interface CustomAnnotation{  
    int value1();  
    String value2();  
    String value3();  
}
```

Section 215.3: Creating and Using Custom Annotations

For creating custom annotations we need to decide

- Target - on which these annotations will work on like field level, method level, type level etc.
- Retention - to what level annotation will be available.

For this, we have built in custom annotations. Check out these mostly used ones:

@Target

Element Types	Where the annotation can be applied
TYPE	class, interface or enumeration
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variables
ANNOTATION_TYPE	annotation type
PARAMETER	parameter

Element Types	Where the annotation can be applied
TYPE	class, interface or enumeration
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variables
ANNOTATION_TYPE	annotation type
PARAMETER	parameter

@Retention

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, discarded during compilation. It will not be available in the compiled class.
RetentionPolicy.CLASS	refers to the .class file, available to java compiler but not to JVM . It is included in the class file.
RetentionPolicy.RUNTIME	refers to the runtime, available to java compiler and JVM .

@Retention

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, discarded during compilation. It will not be available in the compiled class.
RetentionPolicy.CLASS	refers to the .class file, available to java compiler but not to JVM . It is included in the class file.
RetentionPolicy.RUNTIME	refers to the runtime, available to java compiler and JVM .

创建自定义注解

```
@Retention(RetentionPolicy.SOURCE) // 在编译后的类中不可用
@Target(ElementType.METHOD) // 仅可应用于方法
@interface CustomAnnotation{
    int value();
}
```

Creating Custom Annotation

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
    int value();
}
```

使用自定义注解

```
class Foo{
    @CustomAnnotation(value = 1) // 将被注解处理器使用
    public void foo(){...}
}
```

在@CustomAnnotation中提供的值将被注解处理器使用，可能用于在编译时生成代码等。

Using Custom Annotation

```
class Foo{
    @CustomAnnotation(value = 1) // will be used by an annotation processor
    public void foo(){...}
}
```

the value provided inside @CustomAnnotation will be consumed by an Annotationprocessor may be to generate code at compile time etc.

第216章：使用SyncAdapter定期同步数据

应用中的同步适配器组件封装了在设备和服务器之间传输数据的任务代码。根据您在应用中提供的调度和触发条件，同步适配器框架会运行同步适配器组件中的代码。

最近我在研究SyncAdapter，想与大家分享我的知识，可能对其他人有帮助。

第216.1节：每分钟请求服务器值的同步适配器

```
<provider
    android:name=".DummyContentProvider"
    android:authorities="sample.map.com.ipsyncadapter"
    android:exported="false" />

<!-- 此服务实现了我们的SyncAdapter。它需要被导出，以便系统
同步框架可以访问它。 -->
<service android:name=".SyncService"
    android:exported="true">
    <!-- 这个意图过滤器是必需的。它允许系统根据需要启动我们的同步服务。 -->

    <intent-filter>
        <action android:name="android.content.SyncAdapter" />
    </intent-filter>
    <!-- 这指向一个必需的XML文件，描述了我们的SyncAdapter。 -->
    <meta-data android:name="android.content.SyncAdapter"
        android:resource="@xml/syncadapter" />
</service>
```

<!-- 这实现了我们将用作SyncAdapter附加点的账户。由于
我们的SyncAdapter不需要验证当前用户（它只是获取公共RSS
源），因此该账户的实现基本为空。

也可以将SyncAdapter附加到另一个包提供的现有账户。在那种情况下，这个元素可以省略。 -->

```
<service android:name=".AuthenticatorService"
    >
    <!-- 系统启动我们的账户服务所需的必备过滤器。 -->
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <!-- 这指向描述我们账户服务的XML文件。 -->
    <meta-data android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

这段代码需要添加到清单文件中

在上述代码中，我们有同步服务 (syncservice)、内容提供者 (contentprovider) 和身份验证服务 (authenticatorservice)。

在应用程序中，我们需要创建 XML 包以添加 syncadapter 和 authenticator XML 文件。authenticator.xml

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="@string/R.String.accountType"
    android:icon="@mipmap/ic_launcher"
```

Chapter 216: SyncAdapter with periodically do sync of data

The sync adapter component in your app encapsulates the code for the tasks that transfer data between the device and a server. Based on the scheduling and triggers you provide in your app, the sync adapter framework runs the code in the sync adapter component.

Recently i worked on SyncAdapter i want share my knowledge with others,it may help others.

Section 216.1: Sync adapter with every min requesting value from server

```
<provider
    android:name=".DummyContentProvider"
    android:authorities="sample.map.com.ipsyncadapter"
    android:exported="false" />

<!-- This service implements our SyncAdapter. It needs to be exported, so that the system
sync framework can access it. -->
<service android:name=".SyncService"
    android:exported="true">
    <!-- This intent filter is required. It allows the system to launch our sync service
as needed. -->
    <intent-filter>
        <action android:name="android.content.SyncAdapter" />
    </intent-filter>
    <!-- This points to a required XML file which describes our SyncAdapter. -->
    <meta-data android:name="android.content.SyncAdapter"
        android:resource="@xml/syncadapter" />
</service>
```

<!-- This implements the account we'll use as an attachment point for our SyncAdapter. Since
our SyncAdapter doesn't need to authenticate the current user (it just fetches a public RSS
feed), this account's implementation is largely empty.

```
It's also possible to attach a SyncAdapter to an existing account provided by another
package. In that case, this element could be omitted here. -->
<service android:name=".AuthenticatorService"
    >
    <!-- Required filter used by the system to launch our account service. -->
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <!-- This points to an XML file which describes our account service. -->
    <meta-data android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

This code need to be add in manifest file

In above code we have the syncservice and contentprovider and authenticatorservice.

In app we need to create the xml package to add syncadapter and authenticator xml files. **authenticator.xml**

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="@string/R.String.accountType"
    android:icon="@mipmap/ic_launcher"
```

```
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"
/>
```

同步适配器

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="@string/R.String.contentAuthority"
    android:accountType="@string/R.String.accountType"
    android:userVisible="true"
    android:allowParallelSyncs="true"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"/>
```

Authenticator

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
        this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse, String s) {
        return null;
    }

    @Override
    public Bundle 添加账户(AccountAuthenticatorResponse 账户认证响应, String s, String s1, String[] 字符串数组, Bundle 包) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle 确认凭证(AccountAuthenticatorResponse 账户认证响应,
Account 账户, Bundle 包) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle 获取认证令牌(AccountAuthenticatorResponse 账户认证响应, Account 账户, String s, Bundle 包) throws NetworkErrorException {
        return null;
    }

    @Override
    public String 获取认证令牌标签(String s) {
        return null;
    }

    @Override
    public Bundle 更新凭证(AccountAuthenticatorResponse 账户认证响应,
```

```
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"
/>
```

syncadapter

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="@string/R.String.contentAuthority"
    android:accountType="@string/R.String.accountType"
    android:userVisible="true"
    android:allowParallelSyncs="true"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"/>
```

Authenticator

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
        this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse, String s) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String s, String s1, String[] strings, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse, Account account, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse accountAuthenticatorResponse, Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String s) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
```

```

Account 账户, String s, Bundle 包) throws NetworkErrorException {
    return null;
}

@Override
public Bundle 是否有特性(AccountAuthenticatorResponse 账户认证响应, Account
账户, String[] 字符串数组) throws NetworkErrorException {
    return null;
}
}

```

认证服务

```

public class AuthenticatorService extends Service {

    private Authenticator 认证器;

    public 认证服务() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}

```

IpDataDBHelper

```

public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";

    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int
version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "(" + COLUMN_ID + " INTEGER PRIMARY
KEY ,"

```

```

Account account, String s, Bundle bundle) throws NetworkErrorException {
    return null;
}

@Override
public Bundle hasFeatures(AccountAuthenticatorResponse accountAuthenticatorResponse, Account
account, String[] strings) throws NetworkErrorException {
    return null;
}
}

```

AuthenticatorService

```

public class AuthenticatorService extends Service {

    private Authenticator authenticator;

    public AuthenticatorService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}

```

IpDataDBHelper

```

public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";

    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int
version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "(" + COLUMN_ID + " INTEGER PRIMARY
KEY ,"

```

```

        +COLUMN_IP + " INTEGER , " + COLUMN_COUNTRY_CODE + " INTEGER , " +
COLUMN_COUNTRY_NAME +
        " TEXT , " + COLUMN_CITY + " TEXT , " + COLUMN_LATITUDE + " INTEGER , " +
COLUMN_LONGITUDE + " INTEGER );
sqLiteDatabase.execSQL(CREATE_TABLE);
    Log.d("SQL",CREATE_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
    onCreate(sqLiteDatabase);
}

public long AddIPData(ContentValues values)
{
SQLiteDatabase sqLiteDatabase =getWritableDatabase();
    long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
    return insertedRow;
}

public Cursor getAllIpData()
{
    String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE
,COLUMN_LONGITUDE};
SQLiteDatabase sqLiteDatabase =getReadableDatabase();
    Cursor cursor = sqLiteDatabase.query(TABLE_IP_DATA,projection,null,null,null,null,null);
    return cursor;
}

public int deleteAllIpData()
{
SQLiteDatabase sqLiteDatabase=getWritableDatabase();
    int rowDeleted=sqLiteDatabase.delete(TABLE_IP_DATA,null,null);
    return rowDeleted;
}
}

```

MainActivity

```

public class MainActivity extends AppCompatActivity {

    private static final String ACCOUNT_TYPE="sample.map.com.ipsyncadapter";
    private static final String AUTHORITY="sample.map.com.ipsyncadapter";
    private static final String ACCOUNT_NAME="Sync";

    public TextView mIp,mCountryCod,mCountryName,mCity,mLatitude,mLongitude;
    CursorAdapter cursorAdapter;
账户 mAccount;
    私有 字符串 TAG=this.getClass().getCanonicalName();
    列表视图 mListview;
    公共 共享偏好设置 mSharedPreferences;
    @重写
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        设置内容视图(R.布局.activity_main);
        mListview = (列表视图) 查找视图通过ID(R.id.list);
        mIp=(文本视图)查找视图通过ID(R.id.txt_ip);
mCountryCod=(文本视图)查找视图通过ID(R.id.txt_country_code);
        mCountryName=(文本视图)查找视图通过ID(R.id.txt_country_name);

```

```

        +COLUMN_IP + " INTEGER , " + COLUMN_COUNTRY_CODE + " INTEGER , " +
COLUMN_COUNTRY_NAME +
        " TEXT , " + COLUMN_CITY + " TEXT , " + COLUMN_LATITUDE + " INTEGER , " +
COLUMN_LONGITUDE + " INTEGER );
    sqLiteDatabase.execSQL(CREATE_TABLE);
    Log.d("SQL",CREATE_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
    onCreate(sqLiteDatabase);
}

public long AddIPData(ContentValues values)
{
SQLiteDatabase sqLiteDatabase =getWritableDatabase();
    long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
    return insertedRow;
}

public Cursor getAllIpData()
{
    String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE
,COLUMN_LONGITUDE};
    SQLiteDatabase sqLiteDatabase =getReadableDatabase();
    Cursor cursor = sqLiteDatabase.query(TABLE_IP_DATA,projection,null,null,null,null,null);
    return cursor;
}

public int deleteAllIpData()
{
    SQLiteDatabase sqLiteDatabase=getWritableDatabase();
    int rowDeleted=sqLiteDatabase.delete(TABLE_IP_DATA,null,null);
    return rowDeleted;
}
}

```

MainActivity

```

public class MainActivity extends AppCompatActivity {

    private static final String ACCOUNT_TYPE="sample.map.com.ipsyncadapter";
    private static final String AUTHORITY="sample.map.com.ipsyncadapter";
    private static final String ACCOUNT_NAME="Sync";

    public TextView mIp,mCountryCod,mCountryName,mCity,mLatitude,mLongitude;
    CursorAdapter cursorAdapter;
    Account mAccount;
    私有 String TAG=this.getClass().getCanonicalName();
    ListView mListview;
    public Sharedpreferences mSharedPreferences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListview = (ListView) findViewById(R.id.list);
        mIp=(TextView)findViewByid(R.id.txt_ip);
        mCountryCod=(TextView)findViewByid(R.id.txt_country_code);
        mCountryName=(TextView)findViewByid(R.id.txt_country_name);

```

```

mCity=(文本视图)查找视图通过ID(R.id.txt_city);
mLatitude=(文本视图)查找视图通过ID(R.id.txt_latitude);
mLongitude=(文本视图)查找视图通过ID(R.id.txt_longitude);
mSharedPreferences=获取共享偏好设置("MyIp",0);

//使用共享偏好设置在文本视图中显示值。
字符串 txtIp=mSharedPreferences.获取字符串("ipAdr","");
字符串 txtCC=mSharedPreferences.获取字符串("CCode","");
字符串 txtCN=mSharedPreferences.获取字符串("CName","");
字符串 txtC=mSharedPreferences.获取字符串("City","");
字符串 txtLP=mSharedPreferences.获取字符串("Latitude","");
字符串 txtLN=mSharedPreferences.获取字符串("Longitude","");

mIp.设置文本(txtIp);
mCountryCod.设置文本(txtCC);
mCountryName.设置文本(txtCN);
mCity.设置文本(txtC);
mLatitude.setText(txtLP);
mLongitude.setText(txtLN);

mAccount=createSyncAccount(this);
//在此代码中，我使用内容提供者来保存数据。
/* 游标
cursor=getContentResolver().query(MyIPContentProvider.CONTENT_URI,null,null,null,null);
cursorAdapter=new SimpleCursorAdapter(this,R.layout.list_item,cursor,new String
[]{"ip","country_code","country_name","city","latitude","longitude"},
new int[]
{R.id.txt_ip,R.id.txt_country_code,R.id.txt_country_name,R.id.txt_city,R.id.txt_latitude,R.id.txt_longitude},0);

 mListview.setAdapter(cursorAdapter);
getContentResolver().registerContentObserver(MyIPContentProvider.CONTENT_URI,true,new
StockContentObserver(new Handler()));
*/
Bundle settingBundle=new Bundle();
settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL,true);
settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED,true);
ContentResolver.requestSync(mAccount,AUTHORITY,settingBundle);
ContentResolver.setSyncAutomatically(mAccount,AUTHORITY,true);
ContentResolver.addPeriodicSync(mAccount,AUTHORITY,Bundle.EMPTY,60);
}

private Account createSyncAccount(MainActivity mainActivity) {
    Account account=new Account(ACCOUNT_NAME,ACCOUNT_TYPE);
    AccountManager
accountManager=(AccountManager)mainActivity.getSystemService(ACCOUNT_SERVICE);
    if(accountManager.addAccountExplicitly(account,null,null))
    {

    }else
    {
    }
    return account;
}

private class StockContentObserver extends ContentObserver {
    @Override
    public void onChange(boolean selfChange, Uri uri) {
        Log.d(TAG, "CHANGE OBSERVED AT URI: " + uri);
    }
}

```

```

mCity=(TextView)findViewByViewId(R.id.txt_city);
mLatitude=(TextView)findViewByViewId(R.id.txt_latitude);
mLongitude=(TextView)findViewByViewId(R.id.txt_longitude);
mSharedPreferences=getSharedPreferences("MyIp",0);

//Using shared preference iam displaying values in text view.
String txtIp=mSharedPreferences.getString("ipAdr","");
String txtCC=mSharedPreferences.getString("CCode","");
String txtCN=mSharedPreferences.getString("CName","");
String txtC=mSharedPreferences.getString("City","");
String txtLP=mSharedPreferences.getString("Latitude","");
String txtLN=mSharedPreferences.getString("Longitude","");

mIp.setText(txtIp);
mCountryCod.setText(txtCC);
mCountryName.setText(txtCN);
mCity.setText(txtC);
mLatitude.setText(txtLP);
mLongitude.setText(txtLN);

mAccount=createSyncAccount(this);
//In this code i am using content provider to save data.
/* Cursor
cursor=getContentResolver().query(MyIPContentProvider.CONTENT_URI,null,null,null,null);
cursorAdapter=new SimpleCursorAdapter(this,R.layout.list_item,cursor,new String
[]{"ip","country_code","country_name","city","latitude","longitude"},
new int[]
{R.id.txt_ip,R.id.txt_country_code,R.id.txt_country_name,R.id.txt_city,R.id.txt_latitude,R.id.txt_longitude},0);

 mListview.setAdapter(cursorAdapter);
getContentResolver().registerContentObserver(MyIPContentProvider.CONTENT_URI,true,new
StockContentObserver(new Handler()));
*/
Bundle settingBundle=new Bundle();
settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL,true);
settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED,true);
ContentResolver.requestSync(mAccount,AUTHORITY,settingBundle);
ContentResolver.setSyncAutomatically(mAccount,AUTHORITY,true);
ContentResolver.addPeriodicSync(mAccount,AUTHORITY,Bundle.EMPTY,60);
}

private Account createSyncAccount(MainActivity mainActivity) {
    Account account=new Account(ACCOUNT_NAME,ACCOUNT_TYPE);
    AccountManager
accountManager=(AccountManager)mainActivity.getSystemService(ACCOUNT_SERVICE);
    if(accountManager.addAccountExplicitly(account,null,null))
    {

    }else
    {
    }
    return account;
}

private class StockContentObserver extends ContentObserver {
    @Override
    public void onChange(boolean selfChange, Uri uri) {
        Log.d(TAG, "CHANGE OBSERVED AT URI: " + uri);
    }
}

```

```

cursorAdapter.swapCursor(getContentResolver().query(MyIPContentProvider.CONTENT_URI, null, null,
null, null));
}

public StockContentObserver(Handler handler) {
    super(handler);
}

@Override
protected void onResume() {
    super.onResume();
registerReceiver(syncStaredReceiver, new IntentFilter(SyncAdapter.SYNC_STARTED));
registerReceiver(syncFinishedReceiver, new IntentFilter(SyncAdapter.SYNC_FINISHED));
}

@Override
protected void onPause() {
    super.onPause();
unregisterReceiver(syncStaredReceiver);
unregisterReceiver(syncFinishedReceiver);
}
private BroadcastReceiver syncFinishedReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "同步完成！");
        Toast.makeText(getApplicationContext(), "同步完成",
Toast.LENGTH_SHORT).show();
    }
};
private BroadcastReceiver syncStaredReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "同步开始！");
        Toast.makeText(getApplicationContext(), "同步开始...",
Toast.LENGTH_SHORT).show();
    }
}

```

MyIPContentProvider

```

public class MyIPContentProvider extends ContentProvider {

public static final int IP_DATA=1;
private static final String AUTHORITY="sample.map.com.ipsyncadapter";
private static final String TABLE_IP_DATA="ip_data";
public static final Uri CONTENT_URI=Uri.parse("content://" + AUTHORITY + '/' + TABLE_IP_DATA);
private static final UriMatcher URI_MATCHER= new UriMatcher(UriMatcher.NO_MATCH);

static
{
URI_MATCHER.addURI(AUTHORITY, TABLE_IP_DATA, IP_DATA);
}

private IpDataDBHelper myDB;

@Override
public boolean onCreate() {

```

```

cursorAdapter.swapCursor(getContentResolver().query(MyIPContentProvider.CONTENT_URI, null, null,
null, null));
}

public StockContentObserver(Handler handler) {
    super(handler);
}

@Override
protected void onResume() {
    super.onResume();
registerReceiver(syncStaredReceiver, new IntentFilter(SyncAdapter.SYNC_STARTED));
registerReceiver(syncFinishedReceiver, new IntentFilter(SyncAdapter.SYNC_FINISHED));
}

@Override
protected void onPause() {
    super.onPause();
unregisterReceiver(syncStaredReceiver);
unregisterReceiver(syncFinishedReceiver);
}
private BroadcastReceiver syncFinishedReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "Sync finished!");
        Toast.makeText(getApplicationContext(), "Sync Finished",
Toast.LENGTH_SHORT).show();
    }
};
private BroadcastReceiver syncStaredReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "Sync started!");
        Toast.makeText(getApplicationContext(), "Sync started...",
Toast.LENGTH_SHORT).show();
    }
}

```

MyIPContentProvider

```

public class MyIPContentProvider extends ContentProvider {

public static final int IP_DATA=1;
private static final String AUTHORITY="sample.map.com.ipsyncadapter";
private static final String TABLE_IP_DATA="ip_data";
public static final Uri CONTENT_URI=Uri.parse("content://" + AUTHORITY + '/' + TABLE_IP_DATA);
private static final UriMatcher URI_MATCHER= new UriMatcher(UriMatcher.NO_MATCH);

static
{
URI_MATCHER.addURI(AUTHORITY, TABLE_IP_DATA, IP_DATA);
}

private IpDataDBHelper myDB;

@Override
public boolean onCreate() {

```

```

myDB=new IpDataDBHelper(getContext(),null,null,1);
    return false;
}

@Nullable
@Override
public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {
    int uriType=URI_MATCHER.match(uri);
    Cursor cursor=null;
    switch (uriType)
    {
        case IP_DATA:
cursor=myDB.getAllIpData();
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URL");
    }
cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}

@Nullable
@Override
public String getType(Uri uri) {
    return null;
}

@Nullable
@Override
public Uri insert(Uri uri, ContentValues contentValues) {
    int uriType=URI_MATCHER.match(uri);
    long id=0;
    switch (uriType)
    {
        case IP_DATA:
id=myDB.AddIPData(contentValues);
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
getContext().getContentResolver().notifyChange(uri,null);
    return Uri.parse(contentValues + "/" + id);
}

@Override
public int delete(Uri uri, String s, String[] strings) {
    int uriType=URI_MATCHER.match(uri);
    int rowsDeleted=0;

    switch (uriType)
    {
        case IP_DATA:
rowsDeleted=myDB.deleteAllIpData();
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
getContext().getContentResolver().notifyChange(uri,null);
    return rowsDeleted;
}

@Override

```

```

myDB=new IpDataDBHelper(getContext(),null,null,1);
    return false;
}

@Nullable
@Override
public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {
    int uriType=URI_MATCHER.match(uri);
    Cursor cursor=null;
    switch (uriType)
    {
        case IP_DATA:
cursor=myDB.getAllIpData();
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URL");
    }
cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}

@Nullable
@Override
public String getType(Uri uri) {
    return null;
}

@Nullable
@Override
public Uri insert(Uri uri, ContentValues contentValues) {
    int uriType=URI_MATCHER.match(uri);
    long id=0;
    switch (uriType)
    {
        case IP_DATA:
id=myDB.AddIPData(contentValues);
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
getContext().getContentResolver().notifyChange(uri,null);
    return Uri.parse(contentValues + "/" + id);
}

@Override
public int delete(Uri uri, String s, String[] strings) {
    int uriType=URI_MATCHER.match(uri);
    int rowsDeleted=0;

    switch (uriType)
    {
        case IP_DATA:
rowsDeleted=myDB.deleteAllIpData();
        break;
    default:
        throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
getContext().getContentResolver().notifyChange(uri,null);
    return rowsDeleted;
}

@Override

```

```
public int update(Uri uri, ContentValues contentValues, String s, String[] strings) {
    return 0;
}
```

SyncAdapter

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {
    ContentResolver mContentResolver;
    Context mContext;
    public static final String SYNC_STARTED="Sync Started";
    public static final String SYNC_FINISHED="Sync Finished";
    private static final String TAG=SyncAdapter.class.getCanonicalName();
    public SharedPreferences mSharedPreferences;

    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
        this.mContext=context;
        mContentResolver=context.getContentResolver();
        Log.i("SyncAdapter", "SyncAdapter");
    }

    @Override
    public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient contentProviderClient, SyncResult syncResult) {

        Intent intent = new Intent(SYNC_STARTED);
        mContext.sendBroadcast(intent);

        Log.i(TAG, "onPerformSync");

        intent = new Intent(SYNC_FINISHED);
        mContext.sendBroadcast(intent);
        mSharedPreferences =mContext.getSharedPreferences("MyIp",0);
        SharedPreferences.Editor editor=mSharedPreferences.edit();

        mContentResolver.delete(MyIPContentProvider.CONTENT_URI,null,null);

        String data="";

        try {
            URL url =new URL("https://freegeoip.net/json/");
            Log.d(TAG, "URL :" +url);
            HttpURLConnection connection=(HttpURLConnection)url.openConnection();
            Log.d(TAG,"Connection :" +connection);
            connection.connect();
            Log.d(TAG,"连接 1:" +connection);
            InputStream inputStream=connection.getInputStream();
            data=getInputData(inputStream);
            Log.d(TAG,"数据 :" +data);

            if (data != null || !data.equals("null")) {
                JSONObject json0bject = new JSONObject(data);

                String ipa = json0bject.getString("ip");
                String country_code = json0bject.getString("country_code");
                String country_name = json0bject.getString("country_name");
                String region_code=json0bject.getString("region_code");
                String region_name=json0bject.getString("region_name");
            }
        }
    }
}
```

```
public int update(Uri uri, ContentValues contentValues, String s, String[] strings) {
    return 0;
}
```

SyncAdapter

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {
    ContentResolver mContentResolver;
    Context mContext;
    public static final String SYNC_STARTED="Sync Started";
    public static final String SYNC_FINISHED="Sync Finished";
    private static final String TAG=SyncAdapter.class.getCanonicalName();
    public SharedPreferences mSharedPreferences;

    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
        this.mContext=context;
        mContentResolver=context.getContentResolver();
        Log.i("SyncAdapter", "SyncAdapter");
    }

    @Override
    public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient contentProviderClient, SyncResult syncResult) {

        Intent intent = new Intent(SYNC_STARTED);
        mContext.sendBroadcast(intent);

        Log.i(TAG, "onPerformSync");

        intent = new Intent(SYNC_FINISHED);
        mContext.sendBroadcast(intent);
        mSharedPreferences =mContext.getSharedPreferences("MyIp",0);
        SharedPreferences.Editor editor=mSharedPreferences.edit();

        mContentResolver.delete(MyIPContentProvider.CONTENT_URI,null,null);

        String data="";

        try {
            URL url =new URL("https://freegeoip.net/json/");
            Log.d(TAG, "URL :" +url);
            HttpURLConnection connection=(HttpURLConnection)url.openConnection();
            Log.d(TAG,"Connection :" +connection);
            connection.connect();
            Log.d(TAG,"Connection 1:" +connection);
            InputStream inputStream=connection.getInputStream();
            data=getInputData(inputStream);
            Log.d(TAG,"Data :" +data);

            if (data != null || !data.equals("null")) {
                JSONObject json0bject = new JSONObject(data);

                String ipa = json0bject.getString("ip");
                String country_code = json0bject.getString("country_code");
                String country_name = json0bject.getString("country_name");
                String region_code=json0bject.getString("region_code");
                String region_name=json0bject.getString("region_name");
            }
        }
    }
}
```

```

String zip_code=jsonObject.getString("zip_code");
String time_zone=jsonObject.getString("time_zone");
String metro_code=jsonObject.getString("metro_code");

String city = jsonObject.getString("city");
String latitude = jsonObject.getString("latitude");
String longitude = jsonObject.getString("longitude");
/* ContentValues values = new ContentValues();
values.put("ip", ipa);
values.put("country_code", country_code);
values.put("country_name", country_name);
values.put("city", city);
values.put("latitude", latitude);
values.put("longitude", longitude);*/
//使用游标适配器显示结果。
//mContentResolver.insert(MyIPContentProvider.CONTENT_URI, values);

//使用共享偏好保存结果。
editor.putString("ipAddr", ipa);
editor.putString("CCode", country_code);
editor.putString("CName", country_name);
editor.putString("City", city);
editor.putString("Latitude", latitude);
editor.putString("Longitude", longitude);
editor.commit();

}
}catch(Exception e){
e.printStackTrace();
}

}

private String getInputData(InputStream inputStream) throws IOException {
StringBuilder builder=new StringBuilder();
BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream));
//String data=null;
/*Log.d(TAG, "Builder 2:"+ bufferedReader.readLine());
while ((data=bufferedReader.readLine())!= null);
{
    builder.append(data);
    Log.d(TAG, "Builder :" +data);
}
Log.d(TAG, "Builder 1 :" +data);
bufferedReader.close();*/
String data=bufferedReader.readLine();
bufferedReader.close();
return data.toString();
}
}

```

SyncService

```

public class SyncService extends Service {
private static SyncAdapter syncAdapter=null;
private static final Object syncAdapterLock=new Object();

@Override
public void onCreate() {
    synchronized (syncAdapterLock)

```

```

String zip_code=jsonObject.getString("zip_code");
String time_zone=jsonObject.getString("time_zone");
String metro_code=jsonObject.getString("metro_code");

String city = jsonObject.getString("city");
String latitude = jsonObject.getString("latitude");
String longitude = jsonObject.getString("longitude");
/* ContentValues values = new ContentValues();
values.put("ip", ipa);
values.put("country_code", country_code);
values.put("country_name", country_name);
values.put("city", city);
values.put("latitude", latitude);
values.put("longitude", longitude);*/
//Using cursor adapter for results.
//mContentResolver.insert(MyIPContentProvider.CONTENT_URI, values);


```

```

//Using Shared preference for results.
editor.putString("ipAddr", ipa);
editor.putString("CCode", country_code);
editor.putString("CName", country_name);
editor.putString("City", city);
editor.putString("Latitude", latitude);
editor.putString("Longitude", longitude);
editor.commit();

}
}catch(Exception e){
e.printStackTrace();
}
}


```

```

private String getInputData(InputStream inputStream) throws IOException {
StringBuilder builder=new StringBuilder();
BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream));
//String data=null;
/*Log.d(TAG, "Builder 2:"+ bufferedReader.readLine());
while ((data=bufferedReader.readLine())!= null);
{
    builder.append(data);
    Log.d(TAG, "Builder :" +data);
}
Log.d(TAG, "Builder 1 :" +data);
bufferedReader.close();*/
String data=bufferedReader.readLine();
bufferedReader.close();
return data.toString();
}
}


```

SyncService

```

public class SyncService extends Service {
private static SyncAdapter syncAdapter=null;
private static final Object syncAdapterLock=new Object();

@Override
public void onCreate() {
    synchronized (syncAdapterLock)

```

```
{  
    if(syncAdapter==null)  
    {  
        syncAdapter =new SyncAdapter(getApplicationContext(),true);  
    }  
}  
  
@Nullable  
@Override  
public IBinder onBind(Intent intent) {  
    return syncAdapter.getSyncAdapterBinder();  
}
```

```
{  
    if(syncAdapter==null)  
    {  
        syncAdapter =new SyncAdapter(getApplicationContext(),true);  
    }  
}  
  
@Nullable  
@Override  
public IBinder onBind(Intent intent) {  
    return syncAdapter.getSyncAdapterBinder();  
}  
}
```

第217章：Fastjson

Fastjson 是一个Java库，可用于将Java对象转换为其JSON表示形式。它也可以用于将JSON字符串转换为等效的Java对象。

Fastjson的特点：

在服务器端和安卓客户端提供最佳性能

提供简单的 `toJSONObject()` 和 `parseObject()` 方法，用于将Java对象转换为JSON，反之亦然

允许将预先存在的不可修改对象转换为JSON及从JSON转换回来

广泛支持Java泛型

第217.1节：使用Fastjson解析JSON

你可以查看Fastjson库中的示例

编码

```
import com.alibaba.fastjson.JSON;

Group group = new Group();
group.setId(0L);
group.setName("admin");

User guestUser = new User();
guestUser.setId(2L);
guestUser.setName("guest");

User rootUser = new User();
rootUser.setId(3L);
rootUser.setName("root");

group.addUser(guestUser);
group.addUser(rootUser);

String jsonString = JSON.toJSONString(group);

System.out.println(jsonString);
```

输出

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"}, {"id":3,"name":"root"}]}
```

解码

```
String jsonString = ...;
Group group = JSON.parseObject(jsonString, Group.class);
```

Group.java

```
public class Group {

    private Long id;
```

Chapter 217: Fastjson

Fastjson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

Fastjson Features:

Provide best performance in server side and android client

Provide simple `toJSONObject()` and `parseObject()` methods to convert Java objects to JSON and vice-versa

Allow pre-existing unmodifiable objects to be converted to and from JSON

Extensive support of Java Generics

Section 217.1: Parsing JSON with Fastjson

You can look at example in [Fastjson library](#)

Encode

```
import com.alibaba.fastjson.JSON;

Group group = new Group();
group.setId(0L);
group.setName("admin");

User guestUser = new User();
guestUser.setId(2L);
guestUser.setName("guest");

User rootUser = new User();
rootUser.setId(3L);
rootUser.setName("root");

group.addUser(guestUser);
group.addUser(rootUser);

String jsonString = JSON.toJSONString(group);

System.out.println(jsonString);
```

Output

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"}, {"id":3,"name":"root"}]}
```

Decode

```
String jsonString = ...;
Group group = JSON.parseObject(jsonString, Group.class);
```

Group.java

```
public class Group {

    private Long id;
```

```

private String name;
private List<User> users = new ArrayList<User>();

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public List<User> getUsers() {
    return users;
}

public void setUsers(List<User> users) {
    this.users = users;
}

public void addUser(User user) {
    users.add(user);
}
}

```

User.java

```

public class User {

    private Long id;
    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

第217.2节：将Map类型数据转换为JSON字符串

代码

```

private String name;
private List<User> users = new ArrayList<User>();

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public List<User> getUsers() {
    return users;
}

public void setUsers(List<User> users) {
    this.users = users;
}

public void addUser(User user) {
    users.add(user);
}
}

```

User.java

```

public class User {

    private Long id;
    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

Section 217.2: Convert the data of type Map to JSON String

Code

```
Group group = new Group();
group.setId(1);
group.setName("Ke");

User user1 = new User();
user1.setId(2);
user1.setName("Liu");

User user2 = new User();
user2.setId(3);
user2.setName("Yue");
group.getList().add(user1);
group.getList().add(user2);

Map<Integer, Object> map = new HashMap<Integer, Object>();
map.put(1, "No.1");
map.put(2, "No.2");
map.put(3, group.getList());

String jsonString = JSON.toJSONString(map);
System.out.println(jsonString);
```

输出

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"}, {"id":3,"name":"Yue"}]}
```

```
Group group = new Group();
group.setId(1);
group.setName("Ke");

User user1 = new User();
user1.setId(2);
user1.setName("Liu");

User user2 = new User();
user2.setId(3);
user2.setName("Yue");
group.getList().add(user1);
group.getList().add(user2);

Map<Integer, Object> map = new HashMap<Integer, Object>();
map.put(1, "No.1");
map.put(2, "No.2");
map.put(3, group.getList());

String jsonString = JSON.toJSONString(map);
System.out.println(jsonString);
```

Output

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"}, {"id":3,"name":"Yue"}]}
```

第218章：Android中使用org.json处理JSON

第218.1节：创建简单的JSON对象

使用空构造函数创建`JSONObject`，并使用重载的`put()`方法添加字段，该方法支持不同类型的参数：

```
try {
    // 创建一个新的JSONObject实例
    final JSONObject object = new JSONObject();

    // 使用 put 方法可以向 JSONObject 添加一个名称/值对
    object.put("名称", "测试");
    object.put("内容", "Hello World!!!!");
    object.put("年份", 2016);
    object.put("值", 3.23);
    object.put("成员", true);
    object.put("空值", JSONObject.NULL);

    // 调用 JSONObject 的 toString() 方法会返回 JSON 的字符串格式。
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "创建 JSONObject 失败", e);
}
```

生成的 JSON 字符串如下所示：

```
{
    "名称": "测试",
    "内容": "Hello World!!!!",
    "年份": 2016,
    "值": 3.23,
    "成员": true,
    "空值": null
}
```

第218.2节：创建带有null值的JSON字符串

如果你需要生成一个值为`null`的JSON字符串，如下所示：

```
{
    "name": null
}
```

那么你必须使用特殊常量`JSONObject.NULL`。

示例代码：

```
jsonObject.put("name", JSONObject.NULL);
```

第218.3节：向JSONObject添加JSONArray

```
// 创建一个新的JSONArray实例
JSONArray array = new JSONArray();
```

Chapter 218: JSON in Android with org.json

Section 218.1: Creating a simple JSON object

Create the `JSONObject` using the empty constructor and add fields using the `put()` method, which is overloaded so that it can be used with different types:

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject();

    // With put you can add a name/value pair to the JSONObject
    object.put("name", "test");
    object.put("content", "Hello World!!!!");
    object.put("year", 2016);
    object.put("value", 3.23);
    object.put("member", true);
    object.put("null_value", JSONObject.NULL);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

The resulting JSON string looks like this:

```
{
    "name": "test",
    "content": "Hello World!!!!",
    "year": 2016,
    "value": 3.23,
    "member": true,
    "null_value": null
}
```

Section 218.2: Create a JSON String with null value

If you need to produce a JSON string with a value of `null` like this:

```
{
    "name": null
}
```

Then you have to use the special constant `JSONObject.NULL`.

Functioning example:

```
jsonObject.put("name", JSONObject.NULL);
```

Section 218.3: Add JSONArray to JSONObject

```
// Create a new instance of a JSONArray
JSONArray array = new JSONArray();
```

```

// 使用 put() 可以向数组添加一个值。
array.put("ASDF");
array.put("QWERTY");

// 创建一个新的 JSONObject 实例
JSONObject obj = new JSONObject();

try {
    // 将 JSONArray 添加到 JSONObject 中
    obj.put("the_array", array);
} catch (JSONException e) {
    e.printStackTrace();
}

String json = obj.toString();

```

生成的 JSON 字符串如下所示：

```
{
    "the_array": [
        "ASDF",
        "QWERTY"
    ]
}
```

第218.4节：解析简单的JSON对象

考虑以下JSON字符串：

```
{
    "title": "test",
    "content": "Hello World!!!",
    "year": 2016,
    "names" : [
        "Hannah",
        "David",
        "Steve"
    ]
}
```

可以使用以下代码解析此JSON对象：

```

try {
    // 从字符串创建一个新实例
    JSONObject jsonObject = new JSONObject(jsonAsString);
    String title = jsonObject.getString("title");
    String content = jsonObject.getString("content");
    int year = jsonObject.getInt("year");
    JSONArray names = jsonObject.getJSONArray("names"); // 用于字符串对象数组
} catch (JSONException e) {
    Log.w(TAG, "无法解析JSON。错误：" + e.getMessage());
}

```

下面是另一个包含嵌套在JSONObject中的JSONArray的示例：

```
{
    "books": [
        {
            "title": "Android JSON 解析",

```

```

// With put() you can add a value to the array.
array.put("ASDF");
array.put("QWERTY");

// Create a new instance of a JSONObject
JSONObject obj = new JSONObject();

try {
    // Add the JSONArray to the JSONObject
    obj.put("the_array", array);
} catch (JSONException e) {
    e.printStackTrace();
}

String json = obj.toString();

```

The resulting JSON string looks like this:

```
{
    "the_array": [
        "ASDF",
        "QWERTY"
    ]
}
```

Section 218.4: Parse simple JSON object

Consider the following JSON string:

```
{
    "title": "test",
    "content": "Hello World!!!",
    "year": 2016,
    "names" : [
        "Hannah",
        "David",
        "Steve"
    ]
}
```

This JSON object can be parsed using the following code:

```

try {
    // create a new instance from a string
    JSONObject jsonObject = new JSONObject(jsonAsString);
    String title = jsonObject.getString("title");
    String content = jsonObject.getString("content");
    int year = jsonObject.getInt("year");
    JSONArray names = jsonObject.getJSONArray("names"); // for an array of String objects
} catch (JSONException e) {
    Log.w(TAG, "Could not parse JSON. Error: " + e.getMessage());
}

```

Here is another example with a JSONArray nested inside JSONObject:

```
{
    "books": [
        {
            "title": "Android JSON Parsing",

```

```
        "times_sold":186
    }
}
```

可以使用以下代码进行解析：

```
JSONObject root = new JSONObject(booksJson);
JSONArray booksArray = root.getJSONArray("books");
JSONObject firstBook = booksArray.getJSONObject(0);
String title = firstBook.getString("title");
int timesSold = firstBook.getInt("times_sold");
```

第218.5节：检查JSON中字段的存在性

有时检查JSON中某个字段是否存在或缺失是很有用的，以避免代码中出现JSONException异常。

为此，可以使用JSONObject#has(String)方法，如以下示例所示：

示例JSON

```
{
    "name": "James"
}
```

Java 代码

```
String jsonStr = " { \"name\":\"James\" }";
JSONObject json = new JSONObject(jsonStr);
// 检查字段 "name" 是否存在
String name, surname;

// 由于我们的 JSON 中存在字段 "name"，此判断将为真。
if (json.has("name")) {
    name = json.getString("name");
}
else {
    name = "John";
}
// 由于我们的 JSON 中没有字段 "surname"，此判断将为假。
if (json.has("surname")) {
    surname = json.getString("surname");
}
else {
    surname = "Doe";
}

// 这里 name == "James" 且 surname == "Doe"。
```

第218.6节：创建嵌套的JSON对象

要生成嵌套的JSON对象，只需将一个JSON对象添加到另一个中：

```
JSONObject mainObject = new JSONObject();           // 主对象
JSONObject requestObject = new JSONObject();         // 包含的对象

try {
```

```
        "times_sold":186
    }
}
```

This can be parsed with the following code:

```
JSONObject root = new JSONObject(booksJson);
JSONArray booksArray = root.getJSONArray("books");
JSONObject firstBook = booksArray.getJSONObject(0);
String title = firstBook.getString("title");
int timesSold = firstBook.getInt("times_sold");
```

Section 218.5: Check for the existence of fields on JSON

Sometimes it's useful to check if a field is present or absent on your JSON to avoid some JSONException on your code.

To achieve that, use the [JSONObject#has\(String\)](#) or the method, like on the following example:

Sample JSON

```
{
    "name": "James"
}
```

Java code

```
String jsonStr = " { \"name\":\"James\" }";
JSONObject json = new JSONObject(jsonStr);
// Check if the field "name" is present
String name, surname;

// This will be true, since the field "name" is present on our JSON.
if (json.has("name")) {
    name = json.getString("name");
}
else {
    name = "John";
}
// This will be false, since our JSON doesn't have the field "surname".
if (json.has("surname")) {
    surname = json.getString("surname");
}
else {
    surname = "Doe";
}

// Here name == "James" and surname == "Doe".
```

Section 218.6: Create nested JSON object

To produce nested JSON object, you need to simply add one JSON object to another:

```
JSONObject mainObject = new JSONObject();           // Host object
JSONObject requestObject = new JSONObject();         // Included object

try {
```

```

requestObject.put("lastname", lastname);
requestObject.put("phone", phone);
requestObject.put("latitude", lat);
requestObject.put("longitude", lon);
requestObject.put("theme", theme);
requestObject.put("text", message);

mainObject.put("claim", requestObject);
} catch (JSONException e) {
    return "JSON Error";
}

```

现在mainObject包含一个名为claim的键，其值为整个requestObject。

第218.7节：更新JSON中的元素

示例JSON用于更新

```
{
"student": {"name": "Rahul", "lastname": "sharma"},
"marks": {"maths": "88"}
}
```

要更新 JSON 中元素的值，我们需要赋值并更新。

```

try {
    // 创建一个新的 JSONObject 实例
    final JSONObject object = new JSONObject(jsonString);

    JSONObject studentJSON = object.getJSONObject("student");
    studentJSON.put("name", "Kumar");

    object.remove("student");

    object.put("student", studentJSON);

    // 调用 JSONObject 的 toString() 方法会返回 JSON 的字符串格式。
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "创建 JSONObject 失败", e);
}

```

更新后的值

```
{
"student": {"name": "Kumar", "lastname": "sharma"},
"marks": {"maths": "88"}
}
```

第218.8节：使用 JsonReader 从流中读取 JSON

JsonReader 将 JSON 编码的值作为标记流读取。

```

public List<Message> readJsonStream(InputStream in) throws IOException {
    JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
    try {
        return readMessagesArray(reader);
    }
}

```

```

requestObject.put("lastname", lastname);
requestObject.put("phone", phone);
requestObject.put("latitude", lat);
requestObject.put("longitude", lon);
requestObject.put("theme", theme);
requestObject.put("text", message);

mainObject.put("claim", requestObject);
} catch (JSONException e) {
    return "JSON Error";
}

```

Now mainObject contains a key called claim with the whole requestObject as a value.

Section 218.7: Updating the elements in the JSON

sample json to update

```
{
"student": {"name": "Rahul", "lastname": "sharma"},
"marks": {"maths": "88"}
}
```

To update the elements value in the json we need to assign the value and update.

```

try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject(jsonString);

    JSONObject studentJSON = object.getJSONObject("student");
    studentJSON.put("name", "Kumar");

    object.remove("student");

    object.put("student", studentJSON);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}

```

updated value

```
{
"student": {"name": "Kumar", "lastname": "sharma"},
"marks": {"maths": "88"}
}
```

Section 218.8: Using JsonReader to read JSON from a stream

JsonReader reads a JSON encoded value as a stream of tokens.

```

public List<Message> readJsonStream(InputStream in) throws IOException {
    JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
    try {
        return readMessagesArray(reader);
    }
}

```

```

} finally {
    reader.close();
}

public List<Message> readMessagesArray(JsonReader reader) throws IOException {
    List<Message> messages = new ArrayList<Message>();

    reader.beginArray();
    while (reader.hasNext()) {
        messages.add(readMessage(reader));
    }
    reader.endArray();
    return messages;
}

public Message readMessage(JsonReader reader) throws IOException {
    long id = -1;
    String text = null;
    User user = null;
    List<Double> geo = null;

    reader.beginObject();
    while (reader.hasNext()) {
        String name = reader.nextName();
        if (name.equals("id")) {
            id = reader.nextLong();
        } 否则如果 (name.等于("text")) {
            text = reader.nextString();
        } 否则如果 (name.等于("geo") && reader.peek() != JsonToken.NULL) {
            geo = readDoublesArray(reader);
        } 否则如果 (name.等于("user")) {
            user = readUser(reader);
        } else {
            reader.skipValue();
        }
    }
    reader.endObject();
    return new Message(id, text, user, geo);
}

public List<Double> readDoublesArray(JsonReader reader) throws IOException {
    List<Double> doubles = new ArrayList<Double>();

    reader.beginArray();
    while (reader.hasNext()) {
        doubles.add(reader.nextDouble());
    }
    reader.endArray();
    return doubles;
}

public User readUser(JsonReader reader) throws IOException {
    String username = null;
    int followersCount = -1;

    reader.beginObject();
    while (reader.hasNext()) {
        String name = reader.nextName();
        if (name.equals("name")) {
            username = reader.nextString();
        } else if (name.equals("followers_count")) {
}

```

```

} finally {
    reader.close();
}

public List<Message> readMessagesArray(JsonReader reader) throws IOException {
    List<Message> messages = new ArrayList<Message>();

    reader.beginArray();
    while (reader.hasNext()) {
        messages.add(readMessage(reader));
    }
    reader.endArray();
    return messages;
}

public Message readMessage(JsonReader reader) throws IOException {
    long id = -1;
    String text = null;
    User user = null;
    List<Double> geo = null;

    reader.beginObject();
    while (reader.hasNext()) {
        String name = reader.nextName();
        if (name.equals("id")) {
            id = reader.nextLong();
        } else if (name.equals("text")) {
            text = reader.nextString();
        } else if (name.equals("geo") && reader.peek() != JsonToken.NULL) {
            geo = readDoublesArray(reader);
        } else if (name.equals("user")) {
            user = readUser(reader);
        } else {
            reader.skipValue();
        }
    }
    reader.endObject();
    return new Message(id, text, user, geo);
}

public List<Double> readDoublesArray(JsonReader reader) throws IOException {
    List<Double> doubles = new ArrayList<Double>();

    reader.beginArray();
    while (reader.hasNext()) {
        doubles.add(reader.nextDouble());
    }
    reader.endArray();
    return doubles;
}

public User readUser(JsonReader reader) throws IOException {
    String username = null;
    int followersCount = -1;

    reader.beginObject();
    while (reader.hasNext()) {
        String name = reader.nextName();
        if (name.equals("name")) {
            username = reader.nextString();
        } else if (name.equals("followers_count")) {
}

```

```

followersCount = reader.nextInt();
} else {
reader.skipValue();
}
}
reader.endObject();
return new User(username, followersCount);
}

```

第218.9节：解析json时处理null字符串

```

{
    "some_string": null,
    "ather_string": "something"
}

```

如果我们使用这种方式：

```

JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");

```

我们将得到输出：

```
someString = "null";
```

所以我们需要提供以下解决方法：

```

/**
 * 根据
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * 我们需要提供一个解决方案来处理可能为null的json字符串。
* <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key) {
    return optNullableString(jsonObject, key, "");
}

/**
 * 根据
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * 我们需要提供一个解决方案来处理可能为null的json字符串。
* <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key, String fallback) {
    if (jsonObject.isNull(key)) {
        return fallback;
    } else {
        return jsonObject.optString(key, fallback);
    }
}

```

然后调用：

```

JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");

```

我们将得到预期的输出：

```

followersCount = reader.nextInt();
} else {
reader.skipValue();
}
}
reader.endObject();
return new User(username, followersCount);
}

```

Section 218.9: Working with null-string when parsing json

```

{
    "some_string": null,
    "ather_string": "something"
}

```

If we will use this way:

```

JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");

```

We will have output:

```
someString = "null";
```

So we need to provide this workaround:

```

/**
 * According to
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * we need to provide a workaround to opt string from json that can be null.
* <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key) {
    return optNullableString(jsonObject, key, "");
}

/**
 * According to
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * we need to provide a workaround to opt string from json that can be null.
* <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key, String fallback) {
    if (jsonObject.isNull(key)) {
        return fallback;
    } else {
        return jsonObject.optString(key, fallback);
    }
}

```

And then call:

```

JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");

```

And we will have Output as we expected:

```
someString = null; //不是字符串 "null"
someString2 = "";
```

第218.10节：处理JSON响应中的动态键

这是一个如何处理响应中动态键的示例。这里的A和B是动态键，可以是任何内容
响应

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Java 代码

```
// ResponseData 是响应的原始字符串
JSONObject responseDataObj = new JSONObject(responseData);
JSONArray responseArray = responseDataObj.getJSONArray("response");
for (int i = 0; i < responseArray.length(); i++) {
  // Nodes ArrayList<ArrayList<String>> 在全局声明
  nodes = new ArrayList<ArrayList<String>>();
  JSONObject obj = responseArray.getJSONObject(i);
  Iterator keys = obj.keys();
  while(keys.hasNext()) {
    // 循环获取动态键
    String currentDynamicKey = (String)keys.next();
    // 获取动态键的值
    JSONArray currentDynamicValue = obj.getJSONArray(currentDynamicKey);
    int jsonArraySize = currentDynamicValue.length();
    if(jsonArraySize > 0) {
      for (int ii = 0; ii < jsonArraySize; ii++) {
        // NameList ArrayList<String> declared globally
        nameList = new ArrayList<String>();
        if(ii == 0) {
          JSONObject nameObj = currentDynamicValue.getJSONObject(ii);
          String name = nameObj.getString("name");
          System.out.print("Name = " + name);
          // Store name in an array list
          nameList.add(name);
        }
      }
    }
  }
}
```

```
someString = null; //not "null"
someString2 = "";
```

Section 218.10: Handling dynamic key for JSON response

This is an example for how to handle dynamic key for response. Here A and B are dynamic keys it can be anything
Response

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Java code

```
// ResponseData is raw string of response
JSONObject responseDataObj = new JSONObject(responseData);
JSONArray responseArray = responseDataObj.getJSONArray("response");
for (int i = 0; i < responseArray.length(); i++) {
  // Nodes ArrayList<ArrayList<String>> declared globally
  nodes = new ArrayList<ArrayList<String>>();
  JSONObject obj = responseArray.getJSONObject(i);
  Iterator keys = obj.keys();
  while(keys.hasNext()) {
    // Loop to get the dynamic key
    String currentDynamicKey = (String)keys.next();
    // Get the value of the dynamic key
    JSONArray currentDynamicValue = obj.getJSONArray(currentDynamicKey);
    int jsonArraySize = currentDynamicValue.length();
    if(jsonArraySize > 0) {
      for (int ii = 0; ii < jsonArraySize; ii++) {
        // NameList ArrayList<String> declared globally
        nameList = new ArrayList<String>();
        if(ii == 0) {
          JSONObject nameObj = currentDynamicValue.getJSONObject(ii);
          String name = nameObj.getString("name");
          System.out.print("Name = " + name);
          // Store name in an array list
          nameList.add(name);
        }
      }
    }
  }
}
```

```
    }
    nodes.add(nameList);
}
}
```

```
    }
    nodes.add(nameList);
}
}
```

第219章：Gson

Gson 是一个Java库，可用于将Java对象转换为其JSON表示形式。Gson将这两者视为非常重要的设计目标。

Gson特性：

提供简单的 `toJson()` 和 `fromJson()` 方法，用于在Java对象和JSON之间相互转换

允许将预先存在的不可修改对象转换为JSON及从JSON转换回来

广泛支持Java泛型

支持任意复杂的对象（具有深层继承层次和广泛使用的泛型类型）

第219.1节：使用Gson解析JSON

示例展示了如何使用Google的Gson库解析JSON对象。

解析对象：

```
class Robot {  
    //可选 - 该注解允许键名与字段名不同，如果键名与字段名相同则可省略。这也是良好的编码实践，因为它将变量名与服务器  
    //键名解耦。  
  
    @SerializedName("version")  
    private String version;  
  
    @SerializedName("age")  
    private int age;  
  
    @SerializedName("robotName")  
    private String name;  
  
    // 可选：好处是即使json响应中缺少该键，也能设置并保留默认值。原始数据类型不需要此项。  
  
    public Robot{  
        version = "";  
        name = "";  
    }  
}
```

然后在需要解析的地方，使用以下代码：

```
String robotJson = "{  
    \"version\": \"JellyBean\",  
    \"age\": 3,  
    \"robotName\": \"Droid\"\n}";  
  
Gson gson = new Gson();  
Robot robot = gson.fromJson(robotJson, Robot.class);
```

解析列表：

Chapter 219: Gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. Gson considers both of these as very important design goals.

Gson Features:

Provide simple `toJson()` and `fromJson()` methods to convert Java objects to JSON and vice-versa

Allow pre-existing unmodifiable objects to be converted to and from JSON

Extensive support of Java Generics

Support arbitrarily complex objects (with deep inheritance hierarchies and extensive use of generic types)

Section 219.1: Parsing JSON with Gson

The example shows parsing a JSON object using the [Gson library from Google](#).

Parsing objects:

```
class Robot {  
    //OPTIONAL - this annotation allows for the key to be different from the field name, and can be  
    //omitted if key and field name are same . Also this is good coding practice as it decouple your  
    //variable names with server keys name  
    @SerializedName("version")  
    private String version;  
  
    @SerializedName("age")  
    private int age;  
  
    @SerializedName("robotName")  
    private String name;  
  
    // optional : Benefit it allows to set default values and retain them, even if key is missing  
    //from Json response. Not required for primitive data types.  
  
    public Robot{  
        version = "";  
        name = "";  
    }  
}
```

Then where parsing needs to occur, use the following:

```
String robotJson = "  
    \"version\": \"JellyBean\",  
    \"age\": 3,  
    \"robotName\": \"Droid\"  
}";  
  
Gson gson = new Gson();  
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Parsing a list:

在获取 JSON 对象列表时，通常你会想要解析它们并转换成 Java 对象。

我们将尝试转换的 JSON 字符串如下：

```
{  
    "owned_dogs": [  
        {  
            "name": "Ron",  
            "age": 12,  
            "breed": "terrier"  
        },  
        {  
            "name": "鲍勃",  
            "age": 4,  
            "breed": "斗牛犬"  
        },  
        {  
            "name": "约翰尼",  
            "age": 3,  
            "breed": "金毛寻回犬"  
        }  
    ]  
}
```

这个特定的 JSON 数组包含三个对象。在我们的 Java 代码中，我们希望将这些对象映射为 Dog 对象。一个 Dog 对象看起来像这样：

```
private class Dog {  
    public String name;  
    public int age;  
  
    @SerializedName("breed")  
    public String breedName;  
}
```

将 JSON 数组转换为 Dog[]：

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

将 Dog[] 转换为 JSON 字符串：

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

要将 JSON 数组转换为 ArrayList<Dog>，可以执行以下操作：

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();  
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

Type 对象 typeListOfDogs 定义了 Dog 对象列表的结构。GSON 可以使用此类型对象将 JSON 数组映射到正确的值。

另外，将 List<Dog> 转换为 JSON 数组也可以用类似的方法完成。

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

When retrieving a list of JSON objects, often you will want to parse them and convert them into Java objects.

The JSON string that we will try to convert is the following:

```
{  
    "owned_dogs": [  
        {  
            "name": "Ron",  
            "age": 12,  
            "breed": "terrier"  
        },  
        {  
            "name": "Bob",  
            "age": 4,  
            "breed": "bulldog"  
        },  
        {  
            "name": "Johny",  
            "age": 3,  
            "breed": "golden retriever"  
        }  
    ]  
}
```

This particular JSON array contains three objects. In our Java code we'll want to map these objects to Dog objects. A Dog object would look like this:

```
private class Dog {  
    public String name;  
    public int age;  
  
    @SerializedName("breed")  
    public String breedName;  
}
```

To convert the JSON array to a Dog[]:

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Converting a Dog[] to a JSON string:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

To convert the JSON array to an ArrayList<Dog> we can do the following:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();  
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

The Type object typeListOfDogs defines what a list of Dog objects would look like. GSON can use this type object to map the JSON array to the right values.

Alternatively, converting a List<Dog> to a JSON array can be done in a similar manner.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

第219.2节：向 Gson 添加自定义转换器

有时你需要以特定格式序列化或反序列化某些字段，例如你的后端可能使用“YYYY-MM-dd HH:mm”格式表示日期，而你希望你的 POJO 使用 Joda Time 中的 DateTime 类。

为了自动将这些字符串转换为 DateTime 对象，你可以使用自定义转换器。

```
/**  
 * 用于转换Joda {@link DateTime} 对象的Gson序列化/反序列化器。  
 */  
public class DateTimeConverter implements JsonSerializer<DateTime>, JsonDeserializer<DateTime> {  
  
    private final DateTimeFormatter dateTimeFormatter;  
  
    @Inject  
    public DateTimeConverter() {  
        this.dateTimeFormatter = DateTimeFormat.forPattern("YYYY-MM-dd HH:mm");  
    }  
  
    @Override  
    public JsonElement serialize(DateTime src, Type typeOfSrc, JsonSerializationContext context) {  
        return new JsonPrimitive(dateTimeFormatter.print(src));  
    }  
  
    @Override  
    public DateTime deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)  
        throws JsonParseException {  
  
        if (json.getAsString() == null || json.getAsString().isEmpty()) {  
            return null;  
        }  
  
        return dateTimeFormatter.parseDateTime(json.getAsString());  
    }  
}
```

为了让 Gson 使用新创建的转换器，您需要在创建 Gson 对象时进行赋值：

```
DateTimeConverter dateTimeConverter = new DateTimeConverter();  
Gson gson = new GsonBuilder().registerTypeAdapter(DateTime.class, dateTimeConverter)  
.create();  
  
String s = gson.toJson(DateTime.now());  
// 这将以所需格式显示日期
```

为了以该格式反序列化日期，您只需定义一个 DateTime 格式的字段：

```
public class SomePojo {  
    private DateTime someDate;  
}
```

当 Gson 遇到 DateTime 类型的字段时，它将调用您的转换器来反序列化该字段。

第219.3节：使用 Gson 解析 List<String>

方法1

```
Gson gson = new Gson();
```

Section 219.2: Adding a custom Converter to Gson

Sometimes you need to serialize or deserialize some fields in a desired format, for example your backend may use the format "YYYY-MM-dd HH:mm" for dates and you want your POJOS to use the DateTime class in Joda Time.

In order to automatically convert these strings into DateTimes object, you can use a custom converter.

```
/**  
 * Gson serialiser/deserialiser for converting Joda {@link DateTime} objects.  
 */  
public class DateTimeConverter implements JsonSerializer<DateTime>, JsonDeserializer<DateTime> {  
  
    private final DateTimeFormatter dateTimeFormatter;  
  
    @Inject  
    public DateTimeConverter() {  
        this.dateTimeFormatter = DateTimeFormat.forPattern("YYYY-MM-dd HH:mm");  
    }  
  
    @Override  
    public JsonElement serialize(DateTime src, Type typeOfSrc, JsonSerializationContext context) {  
        return new JsonPrimitive(dateTimeFormatter.print(src));  
    }  
  
    @Override  
    public DateTime deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)  
        throws JsonParseException {  
  
        if (json.getAsString() == null || json.getAsString().isEmpty()) {  
            return null;  
        }  
  
        return dateTimeFormatter.parseDateTime(json.getAsString());  
    }  
}
```

To make Gson use the newly created converter you need to assign it when creating the Gson object:

```
DateTimeConverter dateTimeConverter = new DateTimeConverter();  
Gson gson = new GsonBuilder().registerTypeAdapter(DateTime.class, dateTimeConverter)  
.create();  
  
String s = gson.toJson(DateTime.now());  
// this will show the date in the desired format
```

In order to deserialize the date in that format you only have to define a field in the DateTime format:

```
public class SomePojo {  
    private DateTime someDate;  
}
```

When Gson encounters a field of type DateTime, it will call your converter in order to deserialize the field.

Section 219.3: Parsing a List<String> with Gson

Method 1

```
Gson gson = new Gson();
```

```
String json = "[ \"Adam\", \"John\", \"Mary\" ]";  
  
Type type = new TypeToken<List<String>>(){}.getType();  
List<String> members = gson.fromJson(json, type);  
Log.v("Members", members.toString());
```

这对于大多数通用容器类非常有用，因为你无法获取参数化类型的类（即：你不能调用 `List<String>.class`）。

方法 2

```
public class StringList extends ArrayList<String> {  
  
...  
  
List<String> members = gson.fromJson(json, StringList.class);
```

或者，你也可以始终继承你想要的类型，然后传入该类。然而这并不总是最佳实践，因为它会返回一个 `StringList` 类型的对象；

第 219.4 节：将 Gson 添加到你的项目中

```
dependencies {  
    compile 'com.google.code.gson:gson:2.8.1'  
}
```

使用最新版本的 Gson

下面这行代码将在每次编译时编译最新版本的 `gson` 库，你无需更改版本号。

优点：你可以使用最新的功能、速度更快且错误更少。

缺点：这可能会破坏与你代码的兼容性。

```
compile 'com.google.code.gson:gson:+'
```

第219.5节：使用Gson将JSON解析为泛型类对象

假设我们有一个JSON字符串：

```
["first", "second", "third"]
```

我们可以将这个JSON字符串解析为一个String数组：

```
Gson gson = new Gson();  
String jsonArray = "[\"first\", \"second\", \"third\"]";  
String[] strings = gson.fromJson(jsonArray, String[].class);
```

但如果我们想将其解析为一个`List<String>`对象，则必须使用`TypeToken`。

示例如下：

```
Gson gson = new Gson();  
String jsonArray = "[\"first\", \"second\", \"third\"]";  
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>() {}.getType());
```

```
String json = "[ \"Adam\", \"John\", \"Mary\" ]";  
  
Type type = new TypeToken<List<String>>(){}.getType();  
List<String> members = gson.fromJson(json, type);  
Log.v("Members", members.toString());
```

This is useful for most generic container classes, since you can't get the class of a parameterized type (ie: you can't call `List<String>.class`).

Method 2

```
public class StringList extends ArrayList<String> {  
  
...  
  
List<String> members = gson.fromJson(json, StringList.class);
```

Alternatively, you can always subclass the type you want, and then pass in that class. However this isn't always best practice, since it will return to you an object of type `StringList`;

Section 219.4: Adding Gson to your project

```
dependencies {  
    compile 'com.google.code.gson:gson:2.8.1'  
}
```

To use latest version of Gson

The below line will compile latest version of gson library every time you compile, you do not have to change version.

Pros: You can use latest features, speed and less bugs.

Cons: It might break compatibility with your code.

```
compile 'com.google.code.gson:gson:+'
```

Section 219.5: Parsing JSON to Generic Class Object with Gson

Suppose we have a JSON string :

```
["first", "second", "third"]
```

We can parse this JSON string into a `String` array :

```
Gson gson = new Gson();  
String jsonArray = "[\"first\", \"second\", \"third\"]";  
String[] strings = gson.fromJson(jsonArray, String[].class);
```

But if we want parse it into a `List<String>` object, we must use `TypeToken`.

Here is the sample :

```
Gson gson = new Gson();  
String jsonArray = "[\"first\", \"second\", \"third\"]";  
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>() {}.getType());
```

假设我们有以下两个类：

```
public class Outer<T> {  
    public int index;  
    public T data;  
}  
  
public class Person {  
    public String firstName;  
    public String lastName;  
}
```

并且我们有一个 JSON 字符串，需要解析为一个Outer<Person>对象。

此示例展示了如何将该 JSON 字符串解析为相关的泛型类对象：

```
String json = .....;  
Type userType = new TypeToken<Outer<Person>>().getType();  
Result<User> userResult = gson.fromJson(json, userType);
```

如果 JSON 字符串需要解析为一个Outer<List<Person>>对象：

```
Type userListType = new TypeToken<Outer<List<Person>>>().getType();  
Result<List<User>> userListResult = gson.fromJson(json, userListType);
```

第219.6节：使用Gson处理继承

Gson默认不支持继承。

假设我们有以下类层次结构：

```
public class BaseClass {  
    int a;  
  
    public int getInt() {  
        return a;  
    }  
}  
  
public class DerivedClass1 extends BaseClass {  
    int b;  
  
    @Override  
    public int getInt() {  
        return b;  
    }  
}  
  
public class DerivedClass2 extends BaseClass {  
    int c;  
  
    @Override  
    public int getInt() {  
        return c;  
    }  
}
```

现在我们想将一个DerivedClass1的实例序列化为JSON字符串

Suppose we have two classes below:

```
public class Outer<T> {  
    public int index;  
    public T data;  
}  
  
public class Person {  
    public String firstName;  
    public String lastName;  
}
```

and we have a JSON string that should be parsed to a Outer<Person> object.

This example shows how to parse this JSON string to the related generic class object:

```
String json = .....;  
Type userType = new TypeToken<Outer<Person>>().getType();  
Result<User> userResult = gson.fromJson(json, userType);
```

If the JSON string should be parsed to a Outer<List<Person>> object:

```
Type userListType = new TypeToken<Outer<List<Person>>>().getType();  
Result<List<User>> userListResult = gson.fromJson(json, userListType);
```

Section 219.6: Using Gson with inheritance

Gson does not support inheritance out of the box.

Let's say we have the following class hierarchy:

```
public class BaseClass {  
    int a;  
  
    public int getInt() {  
        return a;  
    }  
}  
  
public class DerivedClass1 extends BaseClass {  
    int b;  
  
    @Override  
    public int getInt() {  
        return b;  
    }  
}  
  
public class DerivedClass2 extends BaseClass {  
    int c;  
  
    @Override  
    public int getInt() {  
        return c;  
    }  
}
```

And now we want to serialize an instance of DerivedClass1 to a JSON string

```

DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);

```

现在，在另一个地方，我们接收到这个json字符串并想要反序列化它——但在编译时我们只知道它应该是BaseClass的一个实例：

```

BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());

```

但是GSON不知道derivedClass1Json最初是DerivedClass1的一个实例，所以这将打印出10。

如何解决这个问题？

你需要构建你自己的JsonDeserializer来处理这种情况。这个解决方案不是非常完美，但我想不出更好的方法。

首先，在你的基类中添加以下字段

```

@SerializedName("type")
private String typeName;

```

并在基类构造函数中初始化它

```

public BaseClass() {
    typeName = getClass().getName();
}

```

现在添加以下类：

```

public class JsonDeserializerWithInheritance<T> implements JsonDeserializer<T> {

    @Override
    public T deserialize(
        JsonElement json, Type typeOfT, JsonDeserializationContext context)
        throws JsonParseException {
        JsonObject jsonObject = json.getAsJsonObject();
        JsonPrimitive classNamePrimitive = (JsonPrimitive) jsonObject.get("type");

        String className = classNamePrimitive.getAsString();

        Class<?> clazz;
        try {
            clazz = Class.forName(className);
        } catch (ClassNotFoundException e) {
            throw new JsonParseException(e.getMessage());
        }
        return context.deserialize(jsonObject, clazz);
    }
}

```

剩下要做的就是把所有东西连接起来 -

```

GsonBuilder builder = new GsonBuilder();
builder

```

```

DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);

```

Now, in another place, we receive this json string and want to deserialize it - but in compile time we only know it is supposed to be an instance of BaseClass:

```

BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());

```

But GSON does not know derivedClass1Json was originally an instance of DerivedClass1, so this will print out 10.

How to solve this?

You need to build your own JsonDeserializer, that handles such cases. The solution is not perfectly clean, but I could not come up with a better one.

First, add the following field to your base class

```

@SerializedName("type")
private String typeName;

```

And initialize it in the base class constructor

```

public BaseClass() {
    typeName = getClass().getName();
}

```

Now add the following class:

```

public class JsonDeserializerWithInheritance<T> implements JsonDeserializer<T> {

    @Override
    public T deserialize(
        JsonElement json, Type typeOfT, JsonDeserializationContext context)
        throws JsonParseException {
        JsonObject jsonObject = json.getAsJsonObject();
        JsonPrimitive classNamePrimitive = (JsonPrimitive) jsonObject.get("type");

        String className = classNamePrimitive.getAsString();

        Class<?> clazz;
        try {
            clazz = Class.forName(className);
        } catch (ClassNotFoundException e) {
            throw new JsonParseException(e.getMessage());
        }
        return context.deserialize(jsonObject, clazz);
    }
}

```

All there is left to do is hook everything up -

```

GsonBuilder builder = new GsonBuilder();
builder

```

```
.registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
Gson gson = builder.create();
```

现在，运行以下代码 -

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;
String derivedClass1Json = gson.toJson(derivedClass1);

BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

将打印出5。

第219.7节：使用Gson将JSON属性解析为枚举

如果你想用Gson将字符串解析为枚举：

```
{"status": "open"}
```

```
public enum 状态 {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

第219.8节：使用Gson从磁盘加载JSON文件

这将从磁盘加载一个JSON文件并将其转换为指定类型。

```
public static <T> T getFile(String fileName, Class<T> type) throws FileNotFoundException {
    Gson gson = new GsonBuilder()
        .create();
    FileReader json = new FileReader(fileName);
    return gson.fromJson(json, type);
}
```

第219.9节：在Retrofit中使用Gson作为序列化器

首先，你需要在build.gradle文件中添加GsonConverterFactory

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

然后，在创建Retrofit服务时，你必须添加转换工厂：

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
    .baseUrl(someUrl)
    .addConverterFactory(GsonConverterFactory.create(gson))
```

```
.registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
Gson gson = builder.create();
```

And now, running the following code-

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;
String derivedClass1Json = gson.toJson(derivedClass1);

BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Will print out 5.

Section 219.7: Parsing JSON property to enum with Gson

If you want to parse a String to enum with Gson:

```
{"status": "open"}
```

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Section 219.8: Using Gson to load a JSON file from disk

This will load a JSON file from disk and convert it to the given type.

```
public static <T> T getFile(String fileName, Class<T> type) throws FileNotFoundException {
    Gson gson = new GsonBuilder()
        .create();
    FileReader json = new FileReader(fileName);
    return gson.fromJson(json, type);
}
```

Section 219.9: Using Gson as serializer with Retrofit

First of all you need to add the GsonConverterFactory to your build.gradle file

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Then, you have to add the converter factory when creating the Retrofit Service:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
    .baseUrl(someUrl)
    .addConverterFactory(GsonConverterFactory.create(gson))
```

```
.build()  
.create(RetrofitService.class);
```

在创建传递给工厂的 Gson 对象时，您可以添加自定义转换器，从而创建自定义类型转换。

第219.10节：使用 Gson 将 json 数组解析为泛型类

假设我们有一个 json：

```
{  
    "total_count": 132,  
    "page_size": 2,  
    "page_index": 1,  
    "twitter_posts": [  
        {  
            "created_on": 1465935152,  
            "tweet_id": 210462857140252672,  
            "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",  
            "url": "https://twitter.com/twitterapi/status/210462857140252672"  
        },  
        {  
            "created_on": 1465995741,  
            "tweet_id": 735128881808691200,  
            "tweet": "有关推文即将发生的变化的信息现已发布在开发者网站上",  
            "url": "https://twitter.com/twitterapi/status/735128881808691200"  
        }  
    ]  
}
```

我们可以手动将此数组解析为自定义推文（推文列表容器）对象，但使用 fromJson 方法更简单：

```
Gson gson = new Gson();  
String jsonArray = "....";  
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

假设我们有以下两个类：

```
class Tweets {  
    @SerializedName("total_count")  
    int totalCount;  
    @SerializedName("page_size")  
    int pageSize;  
    @SerializedName("page_index")  
    int pageIndex;  
    // 你所需要做的就是定义一个名称正确的 List 变量  
    @SerializedName("twitter_posts")  
    List<Tweet> tweets;  
}  
  
class Tweet {  
    @SerializedName("created_on")  
    long createdOn;  
    @SerializedName("tweet_id")  
    String tweetId;  
    @SerializedName("tweet")  
    String tweetBody;  
    @SerializedName("url")  
}
```

```
.build()  
.create(RetrofitService.class);
```

You can add custom converters when creating the Gson object that you are passing to the factory. Allowing you to create custom type conversions.

Section 219.10: Parsing json array to generic class using Gson

Suppose we have a json：

```
{  
    "total_count": 132,  
    "page_size": 2,  
    "page_index": 1,  
    "twitter_posts": [  
        {  
            "created_on": 1465935152,  
            "tweet_id": 210462857140252672,  
            "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",  
            "url": "https://twitter.com/twitterapi/status/210462857140252672"  
        },  
        {  
            "created_on": 1465995741,  
            "tweet_id": 735128881808691200,  
            "tweet": "Information on the upcoming changes to Tweets is now on the developer site",  
            "url": "https://twitter.com/twitterapi/status/735128881808691200"  
        }  
    ]  
}
```

We can parse this array into a Custom Tweets (tweets list container) object manually, but it is easier to do it with fromJson method:

```
Gson gson = new Gson();  
String jsonArray = "....";  
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Suppose we have two classes below:

```
class Tweets {  
    @SerializedName("total_count")  
    int totalCount;  
    @SerializedName("page_size")  
    int pageSize;  
    @SerializedName("page_index")  
    int pageIndex;  
    // all you need to do is just define List variable with correct name  
    @SerializedName("twitter_posts")  
    List<Tweet> tweets;  
}  
  
class Tweet {  
    @SerializedName("created_on")  
    long createdOn;  
    @SerializedName("tweet_id")  
    String tweetId;  
    @SerializedName("tweet")  
    String tweetBody;  
    @SerializedName("url")  
}
```

```
String url;  
}
```

如果你只需要解析一个json数组，可以在解析时使用以下代码：

```
String tweetsJsonArray = "[{....},{....}]";  
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>() {}.getType());
```

第219.11节：使用Gson的自定义JSON反序列化器

假设你所有响应中的日期都是某种自定义格式，例如/`Date(1465935152)`/，并且你想应用通用规则将所有Json日期反序列化为javaDate实例。在这种情况下，你需要实现自定义Json反序列化器。

json示例：

```
{  
    "id": 1,  
    "created_on": "Date(1465935152)",  
    "updated_on": "Date(1465968945)",  
    "name": "Oleksandr"  
}
```

假设我们有如下类：

```
class 用户 {  
    @SerializedName("id")  
    long id;  
    @SerializedName("created_on")  
    Date 创建时间;  
    @SerializedName("updated_on")  
    Date 更新时间;  
    @SerializedName("name")  
    String 名称;  
}
```

自定义反序列化器：

```
class 日期反序列化器 implements JsonDeserializer<Date> {  
    private static final String 日期前缀 = "/Date(";  
    private static final String 日期后缀 = ")";  
  
    @Override  
    public Date 反序列化(JsonElement json, Type typeOfT, JsonDeserializationContext context)  
throws JsonParseException {  
    String 日期字符串 = json.getAsString();  
    if (日期字符串.startsWith(日期前缀) && 日期字符串.endsWith(日期后缀)) {  
        日期字符串 = 日期字符串.substring(日期前缀.length(), 日期字符串.length() -  
日期后缀.length());  
    } else {  
        throw new JsonParseException("错误的日期格式: " + 日期字符串);  
    }  
    return new Date(Long.parseLong(日期字符串) - TimeZone.getDefault().getRawOffset());  
}
```

用法如下：

```
String url;  
}
```

and if you need just parse a json array you can use this code in your parsing:

```
String tweetsJsonArray = "[{....},{....}]";  
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>() {}.getType());
```

Section 219.11: Custom JSON Deserializer using Gson

Imagine you have all dates in all responses in some custom format, for instance /`Date(1465935152)`/ and you want apply general rule to deserialize all Json dates to java `Date` instances. In this case you need to implement custom Json Deserializer.

Example of json:

```
{  
    "id": 1,  
    "created_on": "Date(1465935152)",  
    "updated_on": "Date(1465968945)",  
    "name": "Oleksandr"  
}
```

Suppose we have this class below:

```
class User {  
    @SerializedName("id")  
    long id;  
    @SerializedName("created_on")  
    Date createdOn;  
    @SerializedName("updated_on")  
    Date updatedOn;  
    @SerializedName("name")  
    String name;  
}
```

Custom deserializer:

```
class DateDeSerializer implements JsonDeserializer<Date> {  
    private static final String DATE_PREFIX = "/Date(";  
    private static final String DATE_SUFFIX = ")";  
  
    @Override  
    public Date deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)  
throws JsonParseException {  
    String dateString = json.getAsString();  
    if (dateString.startsWith(DATE_PREFIX) && dateString.endsWith(DATE_SUFFIX)) {  
        dateString = dateString.substring(DATE_PREFIX.length(), dateString.length() -  
DATE_SUFFIX.length());  
    } else {  
        throw new JsonParseException("Wrong date format: " + dateString);  
    }  
    return new Date(Long.parseLong(dateString) - TimeZone.getDefault().getRawOffset());  
}
```

And the usage:

```
Gson gson = new GsonBuilder()
.registerTypeAdapter(Date.class, new DateDeSerializer())
.create();
String json = "...";
User user = gson.fromJson(json, User.class);
```

使用 Jackson 序列化和反序列化带有 Date 类型的 JSON 字符串

这同样适用于希望使 Gson 的 Date 转换与 Jackson 兼容的情况，例如。

Jackson 通常将 Date 序列化为“自纪元以来的毫秒数”，而 Gson 使用类似 Aug 31, 这样可读的格式 2016 10:26:17 表示日期。这会导致在使用 Gson 反序列化 Jackson 格式的日期时出现 JsonSyntaxExceptions。

为了解决这个问题，你可以添加自定义的序列化器和反序列化器：

```
JsonSerializer<Date> ser = new JsonSerializer<Date>() {
    @Override
    public JsonElement serialize(Date src, Type typeOfSrc, JsonSerializationContext
        context) {
        return src == null ? null : new JsonPrimitive(src.getTime());
    }
};

JsonDeserializer<Date> deser = new JsonDeserializer<Date>() {
    @Override
    public Date deserialize(JsonElement json, Type typeOfT,
        JsonDeserializationContext context) throws JsonParseException {
        return json == null ? null : new Date(json.getAsLong());
    }
};

Gson gson = new GsonBuilder()
.registerTypeAdapter(Date.class, ser)
.registerTypeAdapter(Date.class, deser)
.create();
```

第219.12节：使用 AutoValue 和 Gson 进行 JSON 序列化/反序列化

导入到你的 Gradle 根文件中

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

导入到你的 gradle 应用文件中

```
apt 'com.google.auto.value:auto-value:1.2'
apt 'com.ryanharter.auto.value:auto-value-gson:0.3.1'
provided 'com.jakewharton.auto.value:auto-value-annotations:1.2-update1'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

使用 autovalue 创建对象：

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();
```

```
Gson gson = new GsonBuilder()
.registerTypeAdapter(Date.class, new DateDeSerializer())
.create();
String json = "...";
User user = gson.fromJson(json, User.class);
```

Serialize and deserialize Jackson JSON strings with Date types

This also applies to the case where you want to make Gson Date conversion compatible with Jackson, for example.

Jackson usually serializes Date to "milliseconds since epoch" whereas Gson uses a readable format like Aug 31, 2016 10:26:17 to represent Date. This leads to JsonSyntaxExceptions in Gson when you try to deserialize a Jackson format Date.

To circumvent this, you can add a custom serializer and a custom deserializer:

```
JsonSerializer<Date> ser = new JsonSerializer<Date>() {
    @Override
    public JsonElement serialize(Date src, Type typeOfSrc, JsonSerializationContext
        context) {
        return src == null ? null : new JsonPrimitive(src.getTime());
    }
};

JsonDeserializer<Date> deser = new JsonDeserializer<Date>() {
    @Override
    public Date deserialize(JsonElement json, Type typeOfT,
        JsonDeserializationContext context) throws JsonParseException {
        return json == null ? null : new Date(json.getAsLong());
    }
};

Gson gson = new GsonBuilder()
.registerTypeAdapter(Date.class, ser)
.registerTypeAdapter(Date.class, deser)
.create();
```

Section 219.12: JSON Serialization/Deserialization with AutoValue and Gson

Import in your gradle root file

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Import in your gradle app file

```
apt 'com.google.auto.value:auto-value:1.2'
apt 'com.ryanharter.auto.value:auto-value-gson:0.3.1'
provided 'com.jakewharton.auto.value:auto-value-annotations:1.2-update1'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

Create object with autovalue:

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();
```

```
public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
    return new AutoValue_SignIn.GsonTypeAdapter(gson);
}

public static SignIn create(String signin, String username) {
    return new AutoValue_SignIn(signin, username);
}
```

使用你的 GsonBuilder 创建 Gson 转换器

```
Gson gson = new GsonBuilder()
    .registerTypeAdapterFactory(
        new AutoValueGsonTypeAdapterFactory())
    .create();
```

反序列化

```
String myjsonData = "{\n    \"signin_token\": \"mySigninToken\", \n    \"username\": \"myUsername\" }";\nSignIn signInData = gson.fromJson(myjsonData, Signin.class);
```

序列化

```
Signin myData = SignIn.create("myTokenData", "myUsername");\nString myjsonData = gson.toJson(myData);
```

使用 Gson 是通过 POJO 对象简化序列化和反序列化代码的好方法。其副作用是反射在性能上代价较高。这就是为什么使用 AutoValue-Gson 生成 CustomTypeAdapter 可以避免这种反射开销，同时在 API 发生变化时仍然非常容易更新。

```
public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
    return new AutoValue_SignIn.GsonTypeAdapter(gson);
}

public static SignIn create(String signin, String username) {
    return new AutoValue_SignIn(signin, username);
}
```

Create your Gson converter with your GsonBuilder

```
Gson gson = new GsonBuilder()
    .registerTypeAdapterFactory(
        new AutoValueGsonTypeAdapterFactory())
    .create();
```

Deserialize

```
String myjsonData = "{\n    \"signin_token\": \"mySigninToken\", \n    \"username\": \"myUsername\" }";\nSignIn signInData = gson.fromJson(myjsonData, Signin.class);
```

Serialize

```
Signin myData = SignIn.create("myTokenData", "myUsername");\nString myjsonData = gson.toJson(myData);
```

Using Gson is a great way to simplify Serialization and Deserialization code by using POJO objects. The side effect is that reflection is costly performance wise. That's why using AutoValue-Gson to generate CustomTypeAdapter will avoid this reflection cost while staying very simple to update when an api change is happening.

第220章：Android 架构组件

Android 架构组件是一组新的库，帮助你设计健壮、可测试且可维护的应用程序。主要部分包括：生命周期（Lifecycles）、视图模型（ViewModel）、实时数据（LiveData）和 Room。

第220.1节：在 AppCompatActivity 中使用生命周期

将您的活动从此活动扩展

```
public abstract class BaseCompatLifecycleActivity extends AppCompatActivity implements LifecycleRegistryOwner {
    // 我们需要这个类，因为 LifecycleActivity 继承自 FragmentActivity 而不是 AppCompatActivity

    @NonNull
    private final LifecycleRegistry lifecycleRegistry = new LifecycleRegistry(this);

    @NonNull
    @Override
    public LifecycleRegistry getLifecycle() {
        return lifecycleRegistry;
    }
}
```

第220.2节：添加架构组件

项目 build.gradle

```
allprojects {
    repositories {
        jcenter()
        // 如果使用 Gradle 4.0+，请添加此项
        google()
        // 如果使用 Gradle 版本低于 4.0，请添加此项
        maven { url 'https://maven.google.com' }
    }
}

ext {
    archVersion = '1.0.0-alpha5'
}
```

应用构建 gradle

```
// 用于 Lifecycles、LiveData 和 ViewModel
compile "android.arch.lifecycle:runtime:$archVersion"
compile "android.arch.lifecycle:extensions:$archVersion"
annotationProcessor "android.arch.lifecycle:compiler:$archVersion"

// 用于 Room
compile "android.arch.persistence.room:runtime:$archVersion"
annotationProcessor "android.arch.persistence.room:compiler:$archVersion"

// 用于测试房间迁移
testCompile "android.arch.persistence.room:testing:$archVersion"

// 用于 Room 的 RxJava 支持
```

Chapter 220: Android Architecture Components

Android Architecture Components 是一组新的库，帮助你设计健壮、可测试且可维护的应用程序。主要部分包括：Lifecycle、ViewModel、LiveData、Room。

Section 220.1: Using Lifecycle in AppCompatActivity

Extend your activity from this activity

```
public abstract class BaseCompatLifecycleActivity extends AppCompatActivity implements LifecycleRegistryOwner {
    // We need this class, because LifecycleActivity extends FragmentActivity not AppCompatActivity

    @NonNull
    private final LifecycleRegistry lifecycleRegistry = new LifecycleRegistry(this);

    @NonNull
    @Override
    public LifecycleRegistry getLifecycle() {
        return lifecycleRegistry;
    }
}
```

Section 220.2: Add Architecture Components

Project build.gradle

```
allprojects {
    repositories {
        jcenter()
        // Add this if you use Gradle 4.0+
        google()
        // Add this if you use Gradle < 4.0
        maven { url 'https://maven.google.com' }
    }
}

ext {
    archVersion = '1.0.0-alpha5'
}
```

Application build gradle

```
// For Lifecycles, LiveData, and ViewModel
compile "android.arch.lifecycle:runtime:$archVersion"
compile "android.arch.lifecycle:extensions:$archVersion"
annotationProcessor "android.arch.lifecycle:compiler:$archVersion"

// For Room
compile "android.arch.persistence.room:runtime:$archVersion"
annotationProcessor "android.arch.persistence.room:compiler:$archVersion"

// For testing Room migrations
testCompile "android.arch.persistence.room:testing:$archVersion"

// For Room RxJava support
```

第 220.3 节：带有 LiveData 转换的 ViewModel

```

public class BaseViewModel extends ViewModel {
    private static final int TAG_SEGMENT_INDEX = 2;
    private static final int VIDEOS_LIMIT = 100;

    // 我们在这里保存输入参数
    private final MutableLiveData<Pair<String, String>> urlWithReferrerLiveData = new
    MutableLiveData<>();

    // 将特定的 uri 参数转换为 "tag"
    private final LiveData<String> currentTagLiveData =
    Transformations.map(urlWithReferrerLiveData, pair -> {
        Uri uri = Uri.parse(pair.first);
        List<String> segments = uri.getPathSegments();
        if (segments.size() > TAG_SEGMENT_INDEX)
            return segments.get(TAG_SEGMENT_INDEX);
        return null;
    });

    // 将 "tag" 转换为视频列表
    private final LiveData<List<VideoItem>> videoByTagData =
    Transformations.switchMap(currentTagLiveData, tag -> contentRepository.getVideoByTag(tag,
    VIDEOS_LIMIT));

    ContentRepository contentRepository;

    public BaseViewModel() {
        // 一些初始化
    }

    public void setUrlWithReferrer(String url, String referrer) {
        // 设置值以激活观察者和转换
        urlWithReferrerLiveData.setValue(new Pair<>(url, referrer));
    }

    public LiveData<List<VideoItem>> getVideoByTagData() {
        return videoByTagData;
    }
}

```

界面某处：

```

public class VideoActivity extends BaseCompatActivity {
    private VideoViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 获取 ViewModel
        viewModel = ViewModelProviders.of(this).get(BaseViewModel.class);
        // 添加观察者
        viewModel.getVideoByTagData().observe(this, data -> {
            // 一些检查
            adapter.updateData(data);
        });
    }
}

```

Section 220.3: ViewModel with LiveData transformations

```

public class BaseViewModel extends ViewModel {
    private static final int TAG_SEGMENT_INDEX = 2;
    private static final int VIDEOS_LIMIT = 100;

    // We save input params here
    private final MutableLiveData<Pair<String, String>> urlWithReferrerLiveData = new
    MutableLiveData<>();

    // transform specific uri param to "tag"
    private final LiveData<String> currentTagLiveData =
    Transformations.map(urlWithReferrerLiveData, pair -> {
        Uri uri = Uri.parse(pair.first);
        List<String> segments = uri.getPathSegments();
        if (segments.size() > TAG_SEGMENT_INDEX)
            return segments.get(TAG_SEGMENT_INDEX);
        return null;
    });

    // transform "tag" to videos list
    private final LiveData<List<VideoItem>> videoByTagData =
    Transformations.switchMap(currentTagLiveData, tag -> contentRepository.getVideoByTag(tag,
    VIDEOS_LIMIT));

    ContentRepository contentRepository;

    public BaseViewModel() {
        // some inits
    }

    public void setUrlWithReferrer(String url, String referrer) {
        // set value activates observers and transformations
        urlWithReferrerLiveData.setValue(new Pair<>(url, referrer));
    }

    public LiveData<List<VideoItem>> getVideoByTagData() {
        return videoByTagData;
    }
}

```

Somewhere in UI:

```

public class VideoActivity extends BaseCompatActivity {
    private VideoViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Get ViewModel
        viewModel = ViewModelProviders.of(this).get(BaseViewModel.class);
        // Add observer
        viewModel.getVideoByTagData().observe(this, data -> {
            // some checks
            adapter.updateData(data);
        });
    }
}

```

```

...
if (savedInstanceState == null) {
    // 仅在首次创建时初始化加载
    // 你只需设置参数并
viewModel.setUrlWithReferrer(url, referrer);
}

```

第220.4节：Room持久化

Room需要四个部分：数据库类、DAO类、实体类和迁移类（现在你可以只使用**DDL方法**）：

实体类

```

// 设置自定义表名，添加索引
@Entity(tableName = "videos",
    indices = {@Index("title")})
)
public final class VideoItem {
    @PrimaryKey // 必需的
    public long articleId;
    public String title;
    public String url;
}

// 使用ForeignKey设置表关系
@Entity(tableName = "tags",
    indices = {@Index("score"), @Index("videoId"), @Index("value")},
    foreignKeys = @ForeignKey(entity = VideoItem.class,
        parentColumns = "articleId",
        childColumns = "videoId",
        onDelete = ForeignKey.CASCADE)
)
public final class VideoTag {
    @PrimaryKey
    public long id;
    public long videoId;
    public String displayName;
    public String value;
    public double score;
}

```

DAO 类

```

@Dao
public interface VideoDao {
    // 使用自定义冲突策略创建插入
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void saveVideos(List<VideoItem> videos);

    // 简单更新
    @Update
    void updateVideos(VideoItem... videos);

    @Query("DELETE FROM tags WHERE videoId = :videoId")
    void deleteTagsByVideoId(long videoId);

    // 自定义查询，您可以在这里使用 select/delete
    @Query("SELECT v.* FROM tags t LEFT JOIN videos v ON v.articleId = t.videoId WHERE t.value =

```

```

...
if (savedInstanceState == null) {
    // init loading only at first creation
    // you just set params and
    viewModel.setUrlWithReferrer(url, referrer);
}

```

Section 220.4: Room persistence

Room require four parts: Database class, DAO classes, Entity classes and Migration classes (now you may use **only DDL methods**):

Entity classes

```

// Set custom table name, add indexes
@Entity(tableName = "videos",
    indices = {@Index("title")})
)
public final class VideoItem {
    @PrimaryKey // required
    public long articleId;
    public String title;
    public String url;
}

// Use ForeignKey for setup table relation
@Entity(tableName = "tags",
    indices = {@Index("score"), @Index("videoId"), @Index("value")},
    foreignKeys = @ForeignKey(entity = VideoItem.class,
        parentColumns = "articleId",
        childColumns = "videoId",
        onDelete = ForeignKey.CASCADE)
)
public final class VideoTag {
    @PrimaryKey
    public long id;
    public long videoId;
    public String displayName;
    public String value;
    public double score;
}

```

DAO classes

```

@Dao
public interface VideoDao {
    // Create insert with custom conflict strategy
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void saveVideos(List<VideoItem> videos);

    // Simple update
    @Update
    void updateVideos(VideoItem... videos);

    @Query("DELETE FROM tags WHERE videoId = :videoId")
    void deleteTagsByVideoId(long videoId);

    // Custom query, you may use select/delete here
    @Query("SELECT v.* FROM tags t LEFT JOIN videos v ON v.articleId = t.videoId WHERE t.value =

```

```
:tag ORDER BY updatedAt DESC LIMIT :limit")
LiveData<List<VideoItem>> getVideosByTag(String tag, int limit);
}
```

数据库类

```
// 注册您的实体和 DAO
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

迁移

```
public final class Migrations {
    private static final Migration MIGRATION_1_2 = new Migration(1, 2) {
        @Override
        public void migrate(SupportSQLiteDatabase database) {
            final String[] sqlQueries = {
                "CREATE TABLE IF NOT EXISTS `tags` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, " +
                " `videoId` INTEGER, `displayName` TEXT, `value` TEXT, `score` REAL, " +
                " FOREIGN KEY(`videoId`) REFERENCES `videos`(`articleId`)" +
                " ON UPDATE NO ACTION ON DELETE CASCADE )",
                "CREATE INDEX `index_tags_score` ON `tags`(`score`)",
                "CREATE INDEX `index_tags_videoId` ON `tags`(`videoId`)";
            for (String query : sqlQueries) {
                database.execSQL(query);
            }
        }
    };

    public static final Migration[] ALL = {MIGRATION_1_2};

    private Migrations() {
    }
}
```

在应用程序类中使用或通过 Dagger 提供

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
        .addMigrations(Migrations.ALL).build();
}
```

编写您的仓库：

```
public final class ContentRepository {
    private final ContentDatabase db;
    private final VideoDao videoDao;

    public ContentRepository(ContentDatabase contentDatabase, VideoDao videoDao) {
        this.db = contentDatabase;
        this.videoDao = videoDao;
    }

    public LiveData<List<VideoItem>> getVideoByTag(@Nullable String tag, int limit) {
        // 你可以从网络获取，保存到数据库
        ...
        return videoDao.getVideosByTag(tag, limit);
    }
}
```

```
:tag ORDER BY updatedAt DESC LIMIT :limit")
LiveData<List<VideoItem>> getVideosByTag(String tag, int limit);
}
```

Database class

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

Migrations

```
public final class Migrations {
    private static final Migration MIGRATION_1_2 = new Migration(1, 2) {
        @Override
        public void migrate(SupportSQLiteDatabase database) {
            final String[] sqlQueries = {
                "CREATE TABLE IF NOT EXISTS `tags` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, " +
                " `videoId` INTEGER, `displayName` TEXT, `value` TEXT, `score` REAL, " +
                " FOREIGN KEY(`videoId`) REFERENCES `videos`(`articleId`)" +
                " ON UPDATE NO ACTION ON DELETE CASCADE )",
                "CREATE INDEX `index_tags_score` ON `tags`(`score`)",
                "CREATE INDEX `index_tags_videoId` ON `tags`(`videoId`)";
            for (String query : sqlQueries) {
                database.execSQL(query);
            }
        }
    };

    public static final Migration[] ALL = {MIGRATION_1_2};

    private Migrations() {
    }
}
```

Use in Application class or provide via Dagger

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
        .addMigrations(Migrations.ALL).build();
}
```

Write your repository:

```
public final class ContentRepository {
    private final ContentDatabase db;
    private final VideoDao videoDao;

    public ContentRepository(ContentDatabase contentDatabase, VideoDao videoDao) {
        this.db = contentDatabase;
        this.videoDao = videoDao;
    }

    public LiveData<List<VideoItem>> getVideoByTag(@Nullable String tag, int limit) {
        // you may fetch from network, save to database
        ...
        return videoDao.getVideosByTag(tag, limit);
    }
}
```

}

在 ViewModel 中使用：

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

第220.5节：自定义 LiveData

如果需要自定义逻辑，您可以编写自定义的 LiveData。

如果你只需要转换数据，不要编写自定义类（使用 Transformations 类）

```
public class LocationLiveData extends LiveData<Location> {
    private LocationManager locationManager;

    private LocationListener listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            setValue(location);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // 执行某些操作
        }

        @Override
        public void onProviderEnabled(String provider) {
            // 执行某些操作
        }

        @Override
        public void onProviderDisabled(String provider) {
            // 执行某些操作
        }
    };

    public LocationLiveData(Context context) {
        locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    }

    @Override
    protected void onActive() {
        // 我们有观察者，开始工作
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
    }

    @Override
    protected void onInactive() {
        // 我们没有观察者，停止工作
        locationManager.removeUpdates(listener);
    }
}
```

}

Use in ViewModel:

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

Section 220.5: Custom LiveData

You may write custom LiveData, if you need custom logic.

Don't write custom class, if you only need to transform data (use Transformations class)

```
public class LocationLiveData extends LiveData<Location> {
    private LocationManager locationManager;

    private LocationListener listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            setValue(location);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // Do something
        }

        @Override
        public void onProviderEnabled(String provider) {
            // Do something
        }

        @Override
        public void onProviderDisabled(String provider) {
            // Do something
        }
    };

    public LocationLiveData(Context context) {
        locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    }

    @Override
    protected void onActive() {
        // We have observers, start working
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
    }

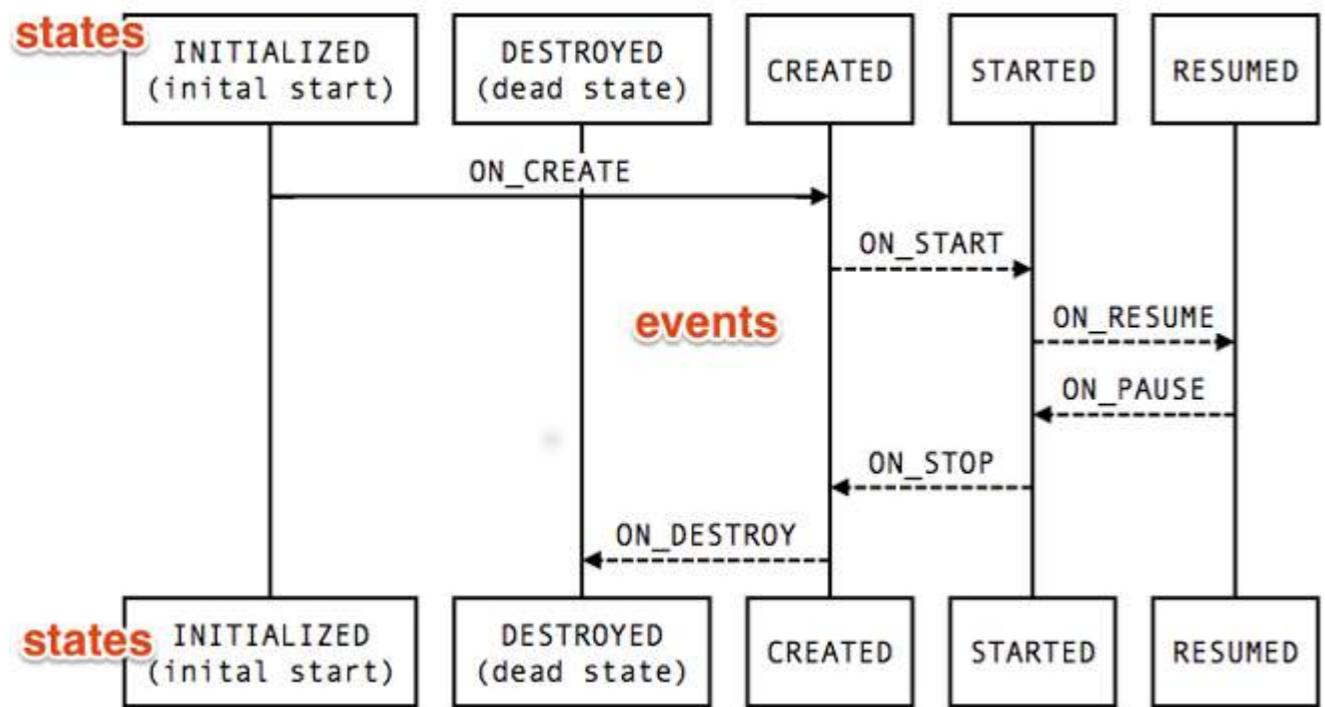
    @Override
    protected void onInactive() {
        // We have no observers, stop working
        locationManager.removeUpdates(listener);
    }
}
```

第220.6节：自定义生命周期感知组件

每个UI组件的生命周期变化如图所示。

Section 220.6: Custom Lifecycle-aware component

Each UI component lifecycle changed as shown at image.



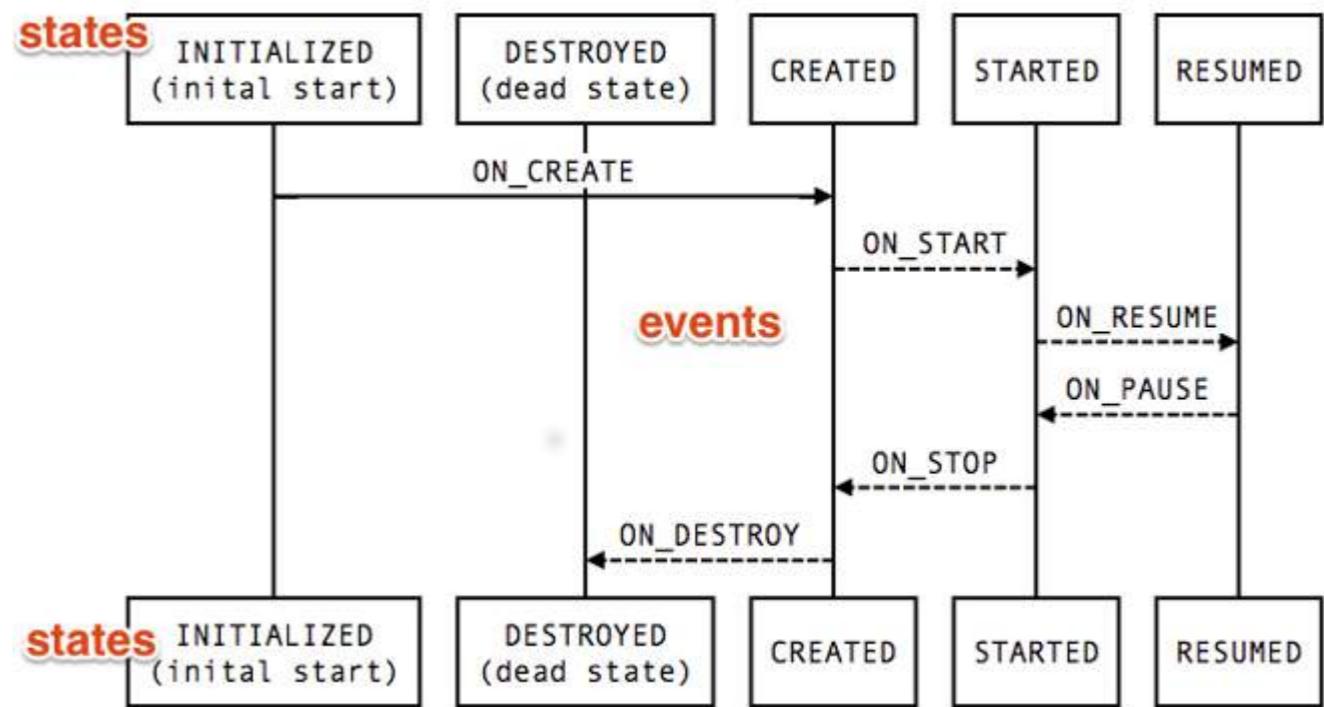
您可以创建一个组件，在生命周期状态变化时收到通知：

```
public class MyLocationListener implements LifecycleObserver {
    private boolean enabled = false;
    private Lifecycle lifecycle;
    public MyLocationListener(Context context, Lifecycle lifecycle, Callback callback) {
        ...
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    void start() {
        if (enabled) {
            // 连接
        }
    }

    public void enable() {
        enabled = true;
        if (lifecycle.getState().isAtLeast(STARTED)) {
            // 如果未连接则连接
        }
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
    void stop() {
        // 如果已连接则断开
    }
}
```



You may create component, that will be notified on lifecycle state change:

```
public class MyLocationListener implements LifecycleObserver {
    private boolean enabled = false;
    private Lifecycle lifecycle;
    public MyLocationListener(Context context, Lifecycle lifecycle, Callback callback) {
        ...
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    void start() {
        if (enabled) {
            // connect
        }
    }

    public void enable() {
        enabled = true;
        if (lifecycle.getState().isAtLeast(STARTED)) {
            // connect if not connected
        }
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
    void stop() {
        // disconnect if connected
    }
}
```

第221章 : Jackson

Jackson 是一个多用途的Java库，用于处理JSON。Jackson旨在为开发者提供快速、正确、轻量且符合人体工学的最佳组合。

Jackson 特性：

多处理模式，协作非常好

不仅支持注解，还支持混合注解

完全支持泛型类型

支持多态类型

第221.1节：完整数据绑定示例

JSON 数据

```
{  
    "name" : { "first" : "Joe", "last" : "Sixpack" },  
    "gender" : "MALE",  
    "verified" : false,  
    "userImage" : "keliuyue"  
}
```

只需两行 Java 代码即可将其转换为 User 实例：

```
ObjectMapper mapper = new ObjectMapper(); // 可重用，全局共享  
User user = mapper.readValue(new File("user.json"), User.class);
```

User.class

```
public class User {  
  
    public enum 性别 {男, 女};  
  
    public static class 姓名 {  
        private String _first, _last;  
  
        public String getFirst() {  
            return _first;  
        }  
  
        public String getLast() {  
            return _last;  
        }  
  
        public void setFirst(String s) {  
            _first = s;  
        }  
  
        public void setLast(String s) {  
            _last = s;  
        }  
    }  
}
```

Chapter 221: Jackson

Jackson is a multi-purpose Java library for processing JSON. Jackson aims to be the best possible combination of fast, correct, lightweight, and ergonomic for developers.

Jackson features :

Multi processing mode, and very good collaboration

Not only annotations, but also mixed annotations

Fully support generic types

Support polymorphic types

Section 221.1: Full Data Binding Example

JSON data

```
{  
    "name" : { "first" : "Joe", "last" : "Sixpack" },  
    "gender" : "MALE",  
    "verified" : false,  
    "userImage" : "keliuyue"  
}
```

It takes two lines of Java to turn it into a User instance:

```
ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally  
User user = mapper.readValue(new File("user.json"), User.class);
```

User.class

```
public class User {  
  
    public enum Gender {MALE, FEMALE};  
  
    public static class Name {  
        private String _first, _last;  
  
        public String getFirst() {  
            return _first;  
        }  
  
        public String getLast() {  
            return _last;  
        }  
  
        public void setFirst(String s) {  
            _first = s;  
        }  
  
        public void setLast(String s) {  
            _last = s;  
        }  
    }  
}
```

```
private Gender _gender;
private Name _name;
private boolean _isVerified;
private byte[] _userImage;

public Name getName() {
    return _name;
}

public boolean isVerified() {
    return _isVerified;
}

public Gender getGender() {
    return _gender;
}

public byte[] getUserImage() {
    return _userImage;
}

public void setName(Name n) {
    _name = n;
}

public void setVerified(boolean b) {
    _isVerified = b;
}

public void setGender(Gender g) {
    _gender = g;
}

public void setUserImage(byte[] b) {
    _userImage = b;
}
```

```
private Gender _gender;
private Name _name;
private boolean _isVerified;
private byte[] _userImage;

public Name getName() {
    return _name;
}

public boolean isVerified() {
    return _isVerified;
}

public Gender getGender() {
    return _gender;
}

public byte[] getUserImage() {
    return _userImage;
}

public void setName(Name n) {
    _name = n;
}

public void setVerified(boolean b) {
    _isVerified = b;
}

public void setGender(Gender g) {
    _gender = g;
}

public void setUserImage(byte[] b) {
    _userImage = b;
}
```

序列化回 JSON 同样非常简单：

```
mapper.writeValue(new File("user-modified.json"), user);
```

Marshalling back to JSON is similarly straightforward:

```
mapper.writeValue(new File("user-modified.json"), user);
```

第222章：智能卡

第222.1节：智能卡发送与接收

关于连接，以下代码片段可帮助您理解：

```
//允许您枚举并与连接的 USB 设备通信。
UsbManager mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
//明确请求权限
final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSION), 0);
HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

UsbDevice device = deviceList.get("//你想操作的设备");
if (device != null) {
    mUsbManager.requestPermission(device, mPermissionIntent);
}
```

现在你必须理解，在Java中通信是使用javax.smartcard包进行的，而该包在Android中不可用，所以请查看这里以了解如何通信或发送/接收APDU（智能卡命令）。

正如上述回答中所说

你不能简单地通过bulk-out端点发送APDU（智能卡命令）并期望通过bulk-in端点接收响应APDU。获取端点请参见以下代码片段：

```
UsbEndpoint epOut = null, epIn = null;
UsbInterface usbInterface;

UsbDeviceConnection connection = mUsbManager.openDevice(device);

for (int i = 0; i < device.getInterfaceCount(); i++) {
    usbInterface = device.getInterface(i);
    connection.claimInterface(usbInterface, true);

    for (int j = 0; j < usbInterface.getEndpointCount(); j++) {
        UsbEndpoint ep = usbInterface.getEndpoint(j);

        if (ep.getType() == UsbConstants.USB_ENDPOINT_XFER_BULK) {
            if (ep.getDirection() == UsbConstants.USB_DIR_OUT) {
                // 从主机到设备
            }
        }
    }
}

epOut = ep;
epIn = ep;
```

现在你已经有了用于发送和接收APDU命令和APDU响应块的批量输入和批量输出端点：

发送命令，请参见以下代码片段：

```
public void write(UsbDeviceConnection connection, UsbEndpoint epOut, byte[] command) {
```

Chapter 222: Smartcard

Section 222.1: Smart card send and receive

For connection, here is a snippet to help you understand:

```
//Allows you to enumerate and communicate with connected USB devices.
UsbManager mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
//Explicitly asking for permission
final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSION), 0);
HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

UsbDevice device = deviceList.get("//the device you want to work with");
if (device != null) {
    mUsbManager.requestPermission(device, mPermissionIntent);
}
```

Now you have to understand that in java the communication takes place using package javax.smartcard which is not available for Android so take a look here for getting an idea as to how you can communicate or send/receive APDU (smartcard command).

Now as told in the answer mentioned above

You cannot simply send an APDU (smartcard command) over the bulk-out endpoint and expect to receive a response APDU over the bulk-in endpoint. For getting the endpoints see the code snippet below :

```
UsbEndpoint epOut = null, epIn = null;
UsbInterface usbInterface;

UsbDeviceConnection connection = mUsbManager.openDevice(device);

for (int i = 0; i < device.getInterfaceCount(); i++) {
    usbInterface = device.getInterface(i);
    connection.claimInterface(usbInterface, true);

    for (int j = 0; j < usbInterface.getEndpointCount(); j++) {
        UsbEndpoint ep = usbInterface.getEndpoint(j);

        if (ep.getType() == UsbConstants.USB_ENDPOINT_XFER_BULK) {
            if (ep.getDirection() == UsbConstants.USB_DIR_OUT) {
                // from host to device
                epOut = ep;
            } else if (ep.getDirection() == UsbConstants.USB_DIR_IN) {
                // from device to host
                epIn = ep;
            }
        }
    }
}
```

Now you have the bulk-in and bulk-out endpoints to send and receive APDU command and APDU response blocks:

For sending commands, see the code snippet below:

```
public void write(UsbDeviceConnection connection, UsbEndpoint epOut, byte[] command) {
```

```

result = new StringBuilder();
connection.bulkTransfer(epOut, command, command.length, TIMEOUT);
    //打印日志时可以使用 result 变量
    for (byte bb : command) {
result.append(String.format(" %02X ", bb));
    }
}

```

接收/读取响应请参见以下代码片段：

```

public int read(UsbDeviceConnection connection, UsbEndpoint epIn) {
result = new StringBuilder();
final byte[] buffer = new byte[epIn.getMaxPacketSize()];
int byteCount = 0;
byteCount = connection.bulkTransfer(epIn, buffer, buffer.length, TIMEOUT);

//打印日志时可以使用 result 变量
if (byteCount >= 0) {
    for (byte bb : buffer) {
result.append(String.format(" %02X ", bb));
    }
}

//接收到的缓冲区内容是 : result.toString()
} else {
    //出现错误, 计数为 : " + byteCount
}

return byteCount;
}

```

现在如果你看到这个答案，这里要发送的第一个命令是：

PC_to_RDR_IccPowerOn 命令用于激活卡片。

你可以通过阅读此处 USB 设备类规范文档的第 6.1.1 节来创建该命令。

现在我们以这个命令为例，比如这里的：62000000000000000000000000000000 你可以这样发送：

```
write(connection, epOut, "620000000000000000000000");
```

成功发送 APDU 命令后，你可以使用以下方式读取响应：

```
read(connection, epIn);
```

并接收类似如下内容

```
80 18000000 00 00 00 00 00 3BBF11008131FE4545504100000000000000000000000000000000F1
```

现在这里代码中接收到的响应将会在代码中read()方法的result变量中

```

result = new StringBuilder();
connection.bulkTransfer(epOut, command, command.length, TIMEOUT);
//For Printing logs you can use result variable
for (byte bb : command) {
    result.append(String.format(" %02X ", bb));
}
}

```

And for receive/ read a response see the code snippet below :

```

public int read(UsbDeviceConnection connection, UsbEndpoint epIn) {
result = new StringBuilder();
final byte[] buffer = new byte[epIn.getMaxPacketSize()];
int byteCount = 0;
byteCount = connection.bulkTransfer(epIn, buffer, buffer.length, TIMEOUT);

//For Printing logs you can use result variable
if (byteCount >= 0) {
    for (byte bb : buffer) {
        result.append(String.format(" %02X ", bb));
    }
}

//Buffer received was : result.toString()
} else {
    //Something went wrong as count was : " + byteCount
}

return byteCount;
}

```

Now if you see this answer here the 1st command to be sent is :

PC_to_RDR_IccPowerOn command to activate the card.

which you can create by reading section 6.1.1 of the USB Device Class Specifications doc here.

Now let's take an example of this command like the one here: [62000000000000000000000000000000](#) How you can send this is :

```
write(connection, epOut, "620000000000000000000000");
```

Now after you have successfully sent the APDU command, you can read the response using :

```
read(connection, epIn);
```

And receive something like

```
80 18000000 00 00 00 00 00 3BBF11008131FE4545504100000000000000000000000000000000F1
```

Now the response received in the code here will be in the result variable of read() method from code

第223章：安全性

第223.1节：验证应用签名 - 防篡改检测

本技术详细说明如何确保您的.apk已使用您的开发者证书签名，并利用证书保持一致且只有您能访问的事实。我们可以将此技术分为3个简单步骤：

- 找到您的开发者证书签名。
- 将您的签名嵌入到应用中的字符串常量中。
- 在运行时检查签名是否与我们嵌入的开发者签名匹配。

代码片段如下：

```
private static final int VALID = 0;
private static final int INVALID = 1;

public static int checkAppSignature(Context context) {

    try {
        PackageInfo packageInfo =
            context.getPackageManager().getPackageInfo(context.getPackageName(),
                PackageManager.GET_SIGNATURES);

        for (Signature signature : packageInfo.signatures) {
            byte[] signatureBytes = signature.toByteArray();

            MessageDigest md = MessageDigest.getInstance("SHA");
            md.update(signature.toByteArray());

            final String currentSignature = Base64.encodeToString(md.digest(), Base64.DEFAULT);

            Log.d("REMOVE_ME", "Include this string as a value for SIGNATURE:" + currentSignature);

            //比较签名
            if (SIGNATURE.equals(currentSignature)){
                return VALID;
            }
        } 捕获 (异常 e) {
            //假设在检查签名时出现问题，但我们让调用者决定如何处理。
        }

        return INVALID;
    }
}
```

Chapter 223: Security

Section 223.1: Verifying App Signature - Tamper Detection

This technique details how to ensure that your .apk has been signed with your developer certificate, and leverages the fact that the certificate remains consistent and that only you have access to it. We can break this technique into 3 simple steps:

- Find your developer certificate signature.
- Embed your signature in a String constant in your app.
- Check that the signature at runtime matches our embedded developer signature.

Here's the code snippet:

```
private static final int VALID = 0;
private static final int INVALID = 1;

public static int checkAppSignature(Context context) {

    try {
        PackageInfo packageInfo =
            context.getPackageManager().getPackageInfo(context.getPackageName(),
                PackageManager.GET_SIGNATURES);

        for (Signature signature : packageInfo.signatures) {
            byte[] signatureBytes = signature.toByteArray();

            MessageDigest md = MessageDigest.getInstance("SHA");
            md.update(signature.toByteArray());

            final String currentSignature = Base64.encodeToString(md.digest(), Base64.DEFAULT);

            Log.d("REMOVE_ME", "Include this string as a value for SIGNATURE:" + currentSignature);

            //compare signatures
            if (SIGNATURE.equals(currentSignature)){
                return VALID;
            }
        } catch (Exception e) {
            //assumes an issue in checking signature., but we let the caller decide on what to do.
        }

        return INVALID;
    }
}
```

第224章：如何安全存储密码

第224.1节：使用AES进行加盐密码加密

本示例使用AES算法对密码进行加密。盐的长度可以达到128位。

我们使用`SecureRandom`类生成盐，将其与密码结合生成一个秘密密钥。所使用的类已存在于Android包中`javax.crypto`和`java.security`。

一旦生成密钥，我们必须将其保存在变量中或存储起来。我们将其存储在共享偏好设置中的值`S_KEY`中。然后，使用`Cipher`类的`doFinal`方法对密码进行加密，前提是该类已在`ENCRYPT_MODE`模式下初始化。接着，将加密后的密码从字节数组转换为字符串，并存储在共享偏好设置中。用于生成加密密码的密钥可以用类似的方式解密密码：

```
public class MainActivity extends AppCompatActivity {
    public static final String PROVIDER = "BC";
    public static final int SALT_LENGTH = 20;
    public static final int IV_LENGTH = 16;
    public static final int PBE_ITERATION_COUNT = 100;

    private static final String RANDOM_ALGORITHM = "SHA1PRNG";
    private static final String HASH_ALGORITHM = "SHA-512";
    private static final String PBE_ALGORITHM = "PBEWithSHA256And256BitAES-CBC-BC";
    private static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";
    public static final String SECRET_KEY_ALGORITHM = "AES";
    private static final String TAG = "EncryptionPassword";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String originalPassword = "ThisIsAndroidStudio%$";
        Log.e(TAG, "originalPassword => " + originalPassword);
        String encryptedPassword = encryptAndStorePassword(originalPassword);
        Log.e(TAG, "encryptedPassword => " + encryptedPassword);
        String decryptedPassword = decryptAndGetPassword();
        Log.e(TAG, "decryptedPassword => " + decryptedPassword);
    }

    private String decryptAndGetPassword() {
        SharedPreferences prefs = getSharedPreferences("pswd", MODE_PRIVATE);
        String encryptedPasswrd = prefs.getString("token", "");
        String passwrd = "";
        if (encryptedPasswrd!=null && !encryptedPasswrd.isEmpty()) {
            try {
                String output = prefs.getString("S_KEY", "");
                byte[] encoded = hexStringToByteArray(output);
                SecretKey aesKey = new SecretKeySpec(encoded, SECRET_KEY_ALGORITHM);
                passwrd = decrypt(aesKey, encryptedPasswrd);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return passwrd;
    }

    public String encryptAndStorePassword(String password) {
```

Chapter 224: How to store passwords securely

Section 224.1: Using AES for salted password encryption

This examples uses the AES algorithm for encrypting passwords. The salt length can be up to 128 bit.

We are using the `SecureRandom` class to generate a salt, which is combined with the password to generate a secret key. The classes used are already existing in Android packages `javax.crypto` and `java.security`.

Once a key is generated, we have to preserve this key in a variable or store it. We are storing it among the shared preferences in the value `S_KEY`. Then, a password is encrypted using the `doFinal` method of the `Cipher` class once it is initialised in `ENCRYPT_MODE`. Next, the encrypted password is converted from a byte array into a string and stored among the shared preferences. The key used to generate an encrypted password can be used to decrypt the password in a similar way:

```
public class MainActivity extends AppCompatActivity {
    public static final String PROVIDER = "BC";
    public static final int SALT_LENGTH = 20;
    public static final int IV_LENGTH = 16;
    public static final int PBE_ITERATION_COUNT = 100;

    private static final String RANDOM_ALGORITHM = "SHA1PRNG";
    private static final String HASH_ALGORITHM = "SHA-512";
    private static final String PBE_ALGORITHM = "PBEWithSHA256And256BitAES-CBC-BC";
    private static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";
    public static final String SECRET_KEY_ALGORITHM = "AES";
    private static final String TAG = "EncryptionPassword";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String originalPassword = "ThisIsAndroidStudio%$";
        Log.e(TAG, "originalPassword => " + originalPassword);
        String encryptedPassword = encryptAndStorePassword(originalPassword);
        Log.e(TAG, "encryptedPassword => " + encryptedPassword);
        String decryptedPassword = decryptAndGetPassword();
        Log.e(TAG, "decryptedPassword => " + decryptedPassword);
    }

    private String decryptAndGetPassword() {
        SharedPreferences prefs = getSharedPreferences("pswd", MODE_PRIVATE);
        String encryptedPasswrd = prefs.getString("token", "");
        String passwrd = "";
        if (encryptedPasswrd!=null && !encryptedPasswrd.isEmpty()) {
            try {
                String output = prefs.getString("S_KEY", "");
                byte[] encoded = hexStringToByteArray(output);
                SecretKey aesKey = new SecretKeySpec(encoded, SECRET_KEY_ALGORITHM);
                passwrd = decrypt(aesKey, encryptedPasswrd);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return passwrd;
    }

    public String encryptAndStorePassword(String password) {
```

```

SharedPreferences.Editor editor = getSharedPreferences("pswd", MODE_PRIVATE).edit();
String encryptedPassword = "";
if (password!=null && !password.isEmpty()) {
    SecretKey secretKey = null;
    try {
secretKey = getSecretKey(password, generateSalt());

        byte[] encoded = secretKey.getEncoded();
        String input = byteArrayToHexString(encoded);
        editor.putString("S_KEY", input);
        encryptedPassword = encrypt(secretKey, password);
    } catch (Exception e) {
e.printStackTrace();
    }
}
editor.putString("token", encryptedPassword);
editor.commit();
}

public static String encrypt(SecretKey secret, String cleartext) throws Exception {
try {
    byte[] iv = generateIv();
    String ivHex = byteArrayToHexString(iv);
    IvParameterSpec ivspec = new IvParameterSpec(iv);

    Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
    encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
    byte[] encryptedText = encryptionCipher.doFinal(cleartext.getBytes("UTF-8"));
    String encryptedHex = byteArrayToHexString(encryptedText);

    return ivHex + encryptedHex;
} 捕获 (异常 e) {
Log.e("SecurityException", e.getCause().getLocalizedMessage());
    throw new Exception("Unable to encrypt", e);
}
}

public static String decrypt(SecretKey secret, String encrypted) throws Exception {
try {
Cipher decryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
    String ivHex = encrypted.substring(0, IV_LENGTH * 2);
    String encryptedHex = encrypted.substring(IV_LENGTH * 2);
    IvParameterSpec ivspec = new IvParameterSpec(hexStringToByteArray(ivHex));
    decryptionCipher.init(Cipher.DECRYPT_MODE, secret, ivspec);
    byte[] decryptedText = decryptionCipher.doFinal(hexStringToByteArray(encryptedHex));
    String decrypted = new String(decryptedText, "UTF-8");
    return decrypted;
} 捕获 (异常 e) {
Log.e("SecurityException", e.getCause().getLocalizedMessage());
    throw new Exception("Unable to decrypt", e);
}
}

public static String generateSalt() throws Exception {
try {
    SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
    byte[] salt = new byte[SALT_LENGTH];
random.nextBytes(salt);
    String saltHex = byteArrayToHexString(salt);
    return saltHex;
}
}

```

```

SharedPreferences.Editor editor = getSharedPreferences("pswd", MODE_PRIVATE).edit();
String encryptedPassword = "";
if (password!=null && !password.isEmpty()) {
    SecretKey secretKey = null;
    try {
secretKey = getSecretKey(password, generateSalt());

        byte[] encoded = secretKey.getEncoded();
        String input = byteArrayToHexString(encoded);
        editor.putString("S_KEY", input);
        encryptedPassword = encrypt(secretKey, password);
    } catch (Exception e) {
e.printStackTrace();
    }
}
editor.putString("token", encryptedPassword);
editor.commit();
}

public static String encrypt(SecretKey secret, String cleartext) throws Exception {
try {
    byte[] iv = generateIv();
    String ivHex = byteArrayToHexString(iv);
    IvParameterSpec ivspec = new IvParameterSpec(iv);

    Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
    encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
    byte[] encryptedText = encryptionCipher.doFinal(cleartext.getBytes("UTF-8"));
    String encryptedHex = byteArrayToHexString(encryptedText);

    return ivHex + encryptedHex;
} catch (Exception e) {
Log.e("SecurityException", e.getCause().getLocalizedMessage());
    throw new Exception("Unable to encrypt", e);
}
}

public static String decrypt(SecretKey secret, String encrypted) throws Exception {
try {
Cipher decryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
    String ivHex = encrypted.substring(0, IV_LENGTH * 2);
    String encryptedHex = encrypted.substring(IV_LENGTH * 2);
    IvParameterSpec ivspec = new IvParameterSpec(hexStringToByteArray(ivHex));
    decryptionCipher.init(Cipher.DECRYPT_MODE, secret, ivspec);
    byte[] decryptedText = decryptionCipher.doFinal(hexStringToByteArray(encryptedHex));
    String decrypted = new String(decryptedText, "UTF-8");
    return decrypted;
} catch (Exception e) {
Log.e("SecurityException", e.getCause().getLocalizedMessage());
    throw new Exception("Unable to decrypt", e);
}
}

public static String generateSalt() throws Exception {
try {
    SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
    byte[] salt = new byte[SALT_LENGTH];
random.nextBytes(salt);
    String saltHex = byteArrayToHexString(salt);
    return saltHex;
}
}

```

```

} 捕获 (异常 e) {
    throw new Exception("Unable to generate salt", e);
}

public static String byteArrayToHexString(byte[] b) {
    StringBuffer sb = new StringBuffer(b.length * 2);
    for (int i = 0; i < b.length; i++) {
        int v = b[i] & 0xff;
        if (v < 16) {
            sb.append('0');
        }
    sb.append(Integer.toHexString(v));
    }
    return sb.toString().toUpperCase();
}

public static byte[] hexStringToByteArray(String s) {
    byte[] b = new byte[s.length() / 2];
    for (int i = 0; i < b.length; i++) {
        int index = i * 2;
        int v = Integer.parseInt(s.substring(index, index + 2), 16);
        b[i] = (byte) v;
    }
    return b;
}

public static SecretKey getSecretKey(String password, String salt) throws Exception {
    try {
PBEKeySpec pbeKeySpec = new PBEKeySpec(password.toCharArray(),
hexStringToByteArray(salt), PBE_ITERATION_COUNT, 256);
SecretKeyFactory factory = SecretKeyFactory.getInstance(PBE_ALGORITHM, PROVIDER);
        SecretKey tmp = factory.generateSecret(pbeKeySpec);
        SecretKey secret = new SecretKeySpec(tmp.getEncoded(), SECRET_KEY_ALGORITHM);
        return secret;
    } 捕获 (异常 e) {
        throw new Exception("Unable to get secret key", e);
    }
}

private static byte[] generateIv() throws NoSuchAlgorithmException, NoSuchProviderException {
    SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
    byte[] iv = new byte[IV_LENGTH];
    random.nextBytes(iv);
    return iv;
}

```

```

} catch (Exception e) {
    throw new Exception("Unable to generate salt", e);
}

public static String byteArrayToHexString(byte[] b) {
    StringBuffer sb = new StringBuffer(b.length * 2);
    for (int i = 0; i < b.length; i++) {
        int v = b[i] & 0xff;
        if (v < 16) {
            sb.append('0');
        }
    sb.append(Integer.toHexString(v));
    }
    return sb.toString().toUpperCase();
}

public static byte[] hexStringToByteArray(String s) {
    byte[] b = new byte[s.length() / 2];
    for (int i = 0; i < b.length; i++) {
        int index = i * 2;
        int v = Integer.parseInt(s.substring(index, index + 2), 16);
        b[i] = (byte) v;
    }
    return b;
}

public static SecretKey getSecretKey(String password, String salt) throws Exception {
    try {
PBEKeySpec pbeKeySpec = new PBEKeySpec(password.toCharArray(),
hexStringToByteArray(salt), PBE_ITERATION_COUNT, 256);
        SecretKeyFactory factory = SecretKeyFactory.getInstance(PBE_ALGORITHM, PROVIDER);
        SecretKey tmp = factory.generateSecret(pbeKeySpec);
        SecretKey secret = new SecretKeySpec(tmp.getEncoded(), SECRET_KEY_ALGORITHM);
        return secret;
    } catch (Exception e) {
        throw new Exception("Unable to get secret key", e);
    }
}

private static byte[] generateIv() throws NoSuchAlgorithmException, NoSuchProviderException {
    SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
    byte[] iv = new byte[IV_LENGTH];
    random.nextBytes(iv);
    return iv;
}

```

第225章：安全的SharedPreferences

参数
输入

定义

要加密或解密的字符串值。

Shared Preferences是基于键值对的XML文件。它位于
`/data/data/package_name/shared_prefs/<filename.xml>`路径下。

因此，具有root权限的用户可以导航到此位置并更改其值。如果您想保护Shared Preferences中的值，可以编写一个简单的加密和解密机制。

不过您应该知道，Shared Preferences从来不是为安全设计的，它只是一个简单地数据持久化方式。

第225.1节：确保共享偏好设置

简单编解码器

这里为了说明工作原理，我们可以使用如下简单的加密和解密。

```
public static String encrypt(String input) {
    // 简单加密，强度不高！
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

实现技术

```
public static String pref_name = "My_Shared_Pref";

// 写入操作
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // 如果针对旧设备可使用 commit

// 阅读用
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Chapter 225: Secure SharedPreferences

Parameter

input

Definition

String value to encrypt or decrypt.

Shared Preferences are **key-value based XML files**. It is located under
`/data/data/package_name/shared_prefs/<filename.xml>`.

So a user with root privileges can navigate to this location and can change its values. If you want to protect values in your shared preferences, you can write a simple encryption and decryption mechanism.

You should know tough, that Shared Preferences were never built to be secure, it's just a simple way to persist data.

Section 225.1: Securing a Shared Preference

Simple Codec

Here to illustrate the working principle we can use simple encryption and decryption as follows.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Implementation Technique

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

第226章：安全的SharedPreferences

| 参数 | 定义 |
|----|--------------|
| 输入 | 要加密或解密的字符串值。 |

Shared Preferences是基于键值对的XML文件。它位于
/data/data/package_name/shared_prefs/<filename.xml>目录下。

因此，具有root权限的用户可以导航到此位置并更改其值。如果您想保护Shared Preferences中的值，可以编写一个简单的加密和解密机制。

不过您应该知道，Shared Preferences从来不是为安全设计的，它只是一个简单地数据持久化方式。

第226.1节：保护Shared Preference

简单编解码器

这里为了说明工作原理，我们可以使用如下简单的加密和解密。

```
public static String encrypt(String input) {
    // 简单加密，强度不高！
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

实现技术

```
public static String pref_name = "My_Shared_Pref";

// 写入操作
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // 如果针对旧设备可使用 commit

// 阅读用
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Chapter 226: Secure SharedPreferences

| Parameter | Definition |
|-----------|-------------------------------------|
| input | String value to encrypt or decrypt. |

Shared Preferences are **key-value based XML files**. It is located under
/data/data/package_name/shared_prefs/<filename.xml>.

So a user with root privileges can navigate to this location and can change its values. If you want to protect values in your shared preferences, you can write a simple encryption and decryption mechanism.

You should know tough, that Shared Preferences were never built to be secure, it's just a simple way to persist data.

Section 226.1: Securing a Shared Preference

Simple Codec

Here to illustrate the working principle we can use simple encryption and decryption as follows.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Implementation Technique

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

第227章：SQLite

SQLite是用C语言编写的关系型数据库管理系统。要在Android框架中开始使用SQLite数据库，需要定义一个继承自[SQLiteOpenHelper](#)的类，并根据需要进行自定义。

第227.1节：onUpgrade()方法

[SQLiteOpenHelper](#)是一个用于管理数据库创建和版本管理的辅助类。

在此类中，[onUpgrade\(\)](#)方法负责在您更改schema时升级数据库。当数据库文件已存在但其版本低于当前应用版本中指定的版本时，会调用此方法。对于每个数据库版本，必须应用您所做的具体更改。

```
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    // 当发生升级时，循环遍历每个版本。  
    for (int version = oldVersion + 1; version <= newVersion; version++) {  
        switch (version) {  
  
            case 2:  
                // 应用版本2中所做的更改  
                db.execSQL(  
                    "ALTER TABLE " +  
                    TABLE_PRODUCTS +  
                    " ADD COLUMN " +  
                    COLUMN_DESCRIPTION +  
                    " TEXT;"  
                );  
                break;  
  
            case 3:  
                // 应用版本3中所做的更改  
                db.execSQL(CREATE_TABLE_TRANSACTION);  
                break;  
        }  
    }  
}
```

第227.2节：从Cursor读取数据

下面是一个方法示例，该方法位于[SQLiteOpenHelper](#)子类中。它使用searchTerm字符串过滤结果，遍历Cursor的内容，并将这些内容以List形式返回，列表中包含Product对象。

首先，定义Product POJO类，作为从数据库中检索到的每一行数据的容器：

```
public class Product {  
    long mId;  
    String mName;  
    String mDescription;  
    float mValue;  
    public Product(long id, String name, String description, float value) {  
        mId = id;  
        mName = name;  
        mDescription = description;  
        mValue = value;  
    }  
}
```

Chapter 227: SQLite

SQLite is a relational database management system written in C. To begin working with SQLite databases within the Android framework, define a class that extends [SQLiteOpenHelper](#), and customize as needed.

Section 227.1: onUpgrade() method

[SQLiteOpenHelper](#) is a helper class to manage database creation and version management.

In this class, the [onUpgrade\(\)](#) method is responsible for upgrading the database when you make changes to the schema. It is called when the database file already exists, but its version is lower than the one specified in the current version of the app. For each database version, the specific changes you made have to be applied.

```
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    // Loop through each version when an upgrade occurs.  
    for (int version = oldVersion + 1; version <= newVersion; version++) {  
        switch (version) {  
  
            case 2:  
                // Apply changes made in version 2  
                db.execSQL(  
                    "ALTER TABLE " +  
                    TABLE_PRODUCTS +  
                    " ADD COLUMN " +  
                    COLUMN_DESCRIPTION +  
                    " TEXT;"  
                );  
                break;  
  
            case 3:  
                // Apply changes made in version 3  
                db.execSQL(CREATE_TABLE_TRANSACTION);  
                break;  
        }  
    }  
}
```

Section 227.2: Reading data from a Cursor

Here is an example of a method that would live inside a [SQLiteOpenHelper](#) subclass. It uses the searchTerm String to filter the results, iterates through the Cursor's contents, and returns those contents in a [List](#) of Product Objects.

First, define the Product [POJO class](#) that will be the container for each row retrieved from the database:

```
public class Product {  
    long mId;  
    String mName;  
    String mDescription;  
    float mValue;  
    public Product(long id, String name, String description, float value) {  
        mId = id;  
        mName = name;  
        mDescription = description;  
        mValue = value;  
    }  
}
```

然后，定义一个方法，用于查询数据库，并返回一个包含Product对象的List：

```
public List<Product> searchForProducts(String searchTerm) {  
  
    // 读取数据时应始终获取可读数据库。  
    final SQLiteDatabase database = this.getReadableDatabase();  
  
    final Cursor cursor = database.query(  
        // 要读取的表名  
        TABLE_NAME,  
  
        // 要读取的列的字符串数组  
        new String[]{COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE},  
  
        // 指定读取哪一行的选择参数。  
        // ? 符号是参数。  
        COLUMN_NAME + " LIKE ?",
  
        // 选择的实际参数值，作为字符串数组。  
        // ? 上面的值从这里获取  
        new String[]{"%" + searchTerm + "%"},  
  
        // GroupBy 子句。指定一个列名，将该列中相似的值分组在一起。  
        null,  
  
        // Having 子句。当使用 GroupBy 子句时，允许你指定包含哪些分组。  
        null,  
  
        // OrderBy 子句。指定一个列名，根据该列对结果进行排序。可选地附加 ASC 或 DESC 来指定升序或降序。  
        null  
    );  
  
    // 为了提高性能，首先获取游标中每个列的索引  
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);  
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);  
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);  
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);  
  
    try {  
  
        // 如果 moveToFirst() 返回 false，则游标为空  
        if (!cursor.moveToFirst()) {  
            return new ArrayList<>();  
        }  
  
        final List<Product> products = new ArrayList<>();  
  
        do {  
  
            // 使用上面获取的索引读取表中一行的值  
            final long id = cursor.getLong(idIndex);  
            final String name = cursor.getString(nameIndex);  
            final String description = cursor.getString(descriptionIndex);  
            final float value = cursor.getFloat(valueIndex);  
  
            products.add(new Product(id, name, description, value));  
        } while (cursor.moveToNext());  
    } catch (Exception e) {  
        Log.e("Database", "Error reading products: " + e.getMessage());  
    } finally {  
        cursor.close();  
    }  
}
```

Then, define the method that will query the database, and return a [List](#) of Product Objects:

```
public List<Product> searchForProducts(String searchTerm) {  
  
    // When reading data one should always just get a readable database.  
    final SQLiteDatabase database = this.getReadableDatabase();  
  
    final Cursor cursor = database.query(  
        // Name of the table to read from  
        TABLE_NAME,  
  
        // String array of the columns which are supposed to be read  
        new String[]{COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE},  
  
        // The selection argument which specifies which row is read.  
        // ? symbols are parameters.  
        COLUMN_NAME + " LIKE ?",
  
        // The actual parameters values for the selection as a String array.  
        // ? above take the value from here  
        new String[]{"%" + searchTerm + "%"},  
  
        // GroupBy clause. Specify a column name to group similar values  
        // in that column together.  
        null,  
  
        // Having clause. When using the GroupBy clause this allows you to  
        // specify which groups to include.  
        null,  
  
        // OrderBy clause. Specify a column name here to order the results  
        // according to that column. Optionally append ASC or DESC to specify  
        // an ascending or descending order.  
        null  
    );  
  
    // To increase performance first get the index of each column in the cursor  
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);  
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);  
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);  
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);  
  
    try {  
  
        // If moveToFirst() returns false then cursor is empty  
        if (!cursor.moveToFirst()) {  
            return new ArrayList<>();  
        }  
  
        final List<Product> products = new ArrayList<>();  
  
        do {  
  
            // Read the values of a row in the table using the indexes acquired above  
            final long id = cursor.getLong(idIndex);  
            final String name = cursor.getString(nameIndex);  
            final String description = cursor.getString(descriptionIndex);  
            final float value = cursor.getFloat(valueIndex);  
  
            products.add(new Product(id, name, description, value));  
        } while (cursor.moveToNext());  
    } catch (Exception e) {  
        Log.e("Database", "Error reading products: " + e.getMessage());  
    } finally {  
        cursor.close();  
    }  
}
```

```

    return products;

} finally {
    // 完成后别忘了关闭Cursor以避免内存泄漏。
    // 使用像本例中那样的 try/finally 通常是处理此问题的最佳方式
    cursor.close();

    // 关闭数据库
    database.close();
}
}

```

第227.3节：使用SQLiteOpenHelper类

```

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 3;

    // 所有主键应使用_id作为列名
    public static final String COLUMN_ID = "_id";

    // Products表的表名和列名定义
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Transactions表的表名和列名定义
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // 创建产品表的语句
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + " (" +
        COLUMN_ID + " INTEGER PRIMARY KEY, " +
        COLUMN_DESCRIPTION + " TEXT, " +
        COLUMN_NAME + " TEXT, " +
        COLUMN_VALUE + " REAL" +
    ");";

    // 创建交易表的语句
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " + TABLE_TRANSACTIONS + " (
        " +
        COLUMN_ID + " INTEGER PRIMARY KEY," +
        COLUMN_PRODUCT_ID + " INTEGER," +
        COLUMN_AMOUNT + " INTEGER," +
        " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
        COLUMN_ID + ")" +
    ");";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate 应始终创建您最新的数据库
        // 当应用新安装时调用此方法
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }
}

```

```

    return products;

} finally {
    // Don't forget to close the Cursor once you are done to avoid memory leaks.
    // Using a try/finally like in this example is usually the best way to handle this
    cursor.close();

    // close the database
    database.close();
}
}

```

Section 227.3: Using the SQLiteOpenHelper class

```

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 3;

    // For all Primary Keys _id should be used as column name
    public static final String COLUMN_ID = "_id";

    // Definition of table and column names of Products table
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Definition of table and column names of Transactions table
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // Create Statement for Products Table
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + " (" +
        COLUMN_ID + " INTEGER PRIMARY KEY, " +
        COLUMN_DESCRIPTION + " TEXT, " +
        COLUMN_NAME + " TEXT, " +
        COLUMN_VALUE + " REAL" +
    ");";

    // Create Statement for Transactions Table
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " + TABLE_TRANSACTIONS + " (
        " +
        COLUMN_ID + " INTEGER PRIMARY KEY," +
        COLUMN_PRODUCT_ID + " INTEGER," +
        COLUMN_AMOUNT + " INTEGER," +
        " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
        COLUMN_ID + ")" +
    );";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate should always create your most up to date database
        // This method is called when the app is newly installed
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }
}

```

```

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // onUpgrade 负责在您更改架构时升级数据库// 对于每个版本，必须应用您在该版本中
    // 所做的具体更改。

    for (int version = oldVersion + 1; version <= newVersion; version++) {
        switch (version) {

            案例 2:
            db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
            break;

            case 3:
            db.execSQL(CREATE_TABLE_TRANSACTION);
            break;
        }
    }
}

```

第227.4节：向数据库插入数据

```

// 您需要一个可写数据库来插入数据
final SQLiteDatabase database = openHelper.getWritableDatabase();

// 创建一个 ContentValues 实例，其中包含每列的数据
// 您无需为主键列指定值。
// 这些的唯一值是自动生成的。
final ContentValues values = new ContentValues();
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// 该调用执行更新操作
// 返回值是新行的 rowId 或主键值！
// 如果此方法返回 -1，则插入失败。
final int id = database.insert(
TABLE_NAME, // 将插入数据的表名
null, // 字符串：可选；可以为 null。如果提供的 values 为空，
// 则无法识别列名，且无法插入空行。
// 如果不设置为 null，此参数提供可空列名，
// 用于显式插入 NULL

values // 包含数据的 ContentValues 实例
);

```

第227.5节：批量插入

这里是一次插入大量数据的示例。你想插入的所有数据都集中在一个 ContentValues数组中。

```

@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:

```

```

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // onUpgrade is responsible for upgrading the database when you make
    // changes to the schema. For each version the specific changes you made
    // in that version have to be applied.

    for (int version = oldVersion + 1; version <= newVersion; version++) {
        switch (version) {

            case 2:
                db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
                break;

            case 3:
                db.execSQL(CREATE_TABLE_TRANSACTION);
                break;
        }
    }
}

```

Section 227.4: Insert data into database

```

// You need a writable database to insert data
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the data for each column
// You do not need to specify a value for the PRIMARY KEY column.
// Unique values for these are automatically generated.
final ContentValues values = new ContentValues();
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value is the rowId or primary key value for the new row!
// If this method returns -1 then the insert has failed.
final int id = database.insert(
    TABLE_NAME, // The table name in which the data will be inserted
    null, // String: optional; may be null. If your provided values is empty,
    // no column names are known and an empty row can't be inserted.
    // If not set to null, this parameter provides the name
    // of nullable column name to explicitly insert a NULL

    values // The ContentValues instance which contains the data
);

```

Section 227.5: Bulk insert

Here is an example of inserting large chunks of data at once. All the data you want to insert is gathered inside of a ContentValues array.

```

@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:

```

```

table = IChatContract.MessageColumns.TABLE_NAME;
        break;
    }
mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
    } finally {
    }
mDatabase.setTransactionSuccessful();
    getContext().getContentResolver().notifyChange(uri, null);
    count = values.length;
}
mDatabase.endTransaction();
}
return count;
}

```

下面是一个如何使用它的示例：

```

ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//向 valueList 添加你想要的内容
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);

```

第227.6节：为Android中的SQLite创建合同、辅助类和提供者

DBContract.java

```

//定义本地数据库的表和列
public final class DBContract {
    /*内容授权 (Content Authority) 是内容提供者的名称，使用应用包名作为名称方便在设备上保持唯一性*/
    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";
    //使用 CONTENT_AUTHORITY 创建应用将用来链接内容提供者的所有数据库 URI
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);
    /*URI 的名称，可以与表名相同。
    这将转换为 content://com.yourdomain.yourapp/user/ 作为有效的 URI
    */
    public static final String PATH_USER = "User";
    //为了防止有人意外实例化该契约类,
    //给它一个空的构造函数。
    public DBContract () {}
    //定义用户表的内部类
    public static final class UserEntry implements BaseColumns {
        public static final Uri CONTENT_URI =
        BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();
        public static final String CONTENT_TYPE =
        ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;
    }
}

```

```

table = IChatContract.MessageColumns.TABLE_NAME;
        break;
    }
mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
    } finally {
    }
mDatabase.setTransactionSuccessful();
    getContext().getContentResolver().notifyChange(uri, null);
    count = values.length;
}
mDatabase.endTransaction();
}
return count;
}

```

And here is an example of how to use it:

```

ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);

```

Section 227.6: Create a Contract, Helper and Provider for SQLite in Android

DBContract.java

```

//Define the tables and columns of your local database
public final class DBContract {
    /*Content Authority its a name for the content provider, is convenient to use the package app name to
    be unique on the device */
    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";
    //Use CONTENT_AUTHORITY to create all the database URI's that the app will use to link the
    content provider.
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);
    /*the name of the uri that can be the same as the name of your table.
    this will translate to content://com.yourdomain.yourapp/user/ as a valid URI
    */
    public static final String PATH_USER = "User";
    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public DBContract () {}
    //Intern class that defines the user table
    public static final class UserEntry implements BaseColumns {
        public static final Uri CONTENT_URI =
        BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();
        public static final String CONTENT_TYPE =
        ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;
    }
}

```

```

// 表名
public static final String TABLE_NAME="User";

// 用户表的列
public static final String COLUMN_Name="Name";
public static final String COLUMN_Password="Password";

public static Uri buildUri(long id){
    return ContentUris.withAppendedId(CONTENT_URI,id);
}
}

```

DBHelper.java

```

public class DBHelper extends SQLiteOpenHelper{

// 如果更改数据库的架构，必须增加此数字
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
        mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//创建 users 表
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}
}

```

DBProvider.java

```

public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
}

```

```

//Name of the table
public static final String TABLE_NAME="User";

//Columns of the user table
public static final String COLUMN_Name="Name";
public static final String COLUMN_Password="Password";

public static Uri buildUri(long id){
    return ContentUris.withAppendedId(CONTENT_URI,id);
}
}

```

DBHelper.java

```

public class DBHelper extends SQLiteOpenHelper{

//if you change the schema of the database, you must increment this number
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
        mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//Create the table users
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}
}

```

DBProvider.java

```

public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
}

```

```

private Context mContext;

static final int USER = 100;

static UriMatcher buildUriMatcher() {

    final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
    final String authority = DBContract.CONTENT_AUTHORITY;

    matcher.addURI(authority, DBContract.PATH_USER, USER);

    return matcher;
}

@Override
public boolean onCreate() {
    mDBHelper = new DBHelper(getContext());
    return false;
}

public PeaberryProvider(Context context) {
    mDBHelper = DBHelper.getInstance(context);
    mContext = context;
}

@Override
public String getType(Uri uri) {
    // determine what type of Uri is
    final int match = sUriMatcher.match(uri);

    switch (match) {
        case USER:
            return DBContract.UserEntry.CONTENT_TYPE;

        default:
            throw new UnsupportedOperationException("未知的Uri: " + uri);
    }
}

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
                     String sortOrder) {
    Cursor retCursor;
    try {
        switch (sUriMatcher.match(uri)) {
            case USER:
                retCursor = mDBHelper.getReadableDatabase().query(
                    DBContract.UserEntry.TABLE_NAME,
                    projection,
                    selection,
                    selectionArgs,
                    null,
                    null,
                    sortOrder
                );
                break;
            default:
                throw new UnsupportedOperationException("未知的Uri: " + uri);
        }
    } catch (Exception ex) {
        Log.e("Cursor", ex.toString());
    }
}

```

```

private Context mContext;

static final int USER = 100;

static UriMatcher buildUriMatcher() {

    final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
    final String authority = DBContract.CONTENT_AUTHORITY;

    matcher.addURI(authority, DBContract.PATH_USER, USER);

    return matcher;
}

@Override
public boolean onCreate() {
    mDBHelper = new DBHelper(getContext());
    return false;
}

public PeaberryProvider(Context context) {
    mDBHelper = DBHelper.getInstance(context);
    mContext = context;
}

@Override
public String getType(Uri uri) {
    // determine what type of Uri is
    final int match = sUriMatcher.match(uri);

    switch (match) {
        case USER:
            return DBContract.UserEntry.CONTENT_TYPE;

        default:
            throw new UnsupportedOperationException("Uri unknown: " + uri);
    }
}

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
                     String sortOrder) {
    Cursor retCursor;
    try {
        switch (sUriMatcher.match(uri)) {
            case USER:
                retCursor = mDBHelper.getReadableDatabase().query(
                    DBContract.UserEntry.TABLE_NAME,
                    projection,
                    selection,
                    selectionArgs,
                    null,
                    null,
                    sortOrder
                );
                break;
            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
    } catch (Exception ex) {
        Log.e("Cursor", ex.toString());
    }
}

```

```

    } finally {
mDBHelper.close();
}
return null;
}

@Override
public Uri insert(Uri uri, ContentValues values) {
    final SQLiteDatabase db = mDBHelper.getWritableDatabase();
    final int match = sUriMatcher.match(uri);
Uri returnUri;
try {

    switch (match) {
        case USER: {
            long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
            if (_id > 0)
returnUri = DBContract.UserEntry.buildUri(_id);
            else
                throw new android.database.SQLException("Error at inserting row in " +
uri);
            break;
        }
        default:
            throw new UnsupportedOperationException("未知的Uri: " + uri);
    }
}
mContext.getContentResolver().notifyChange(uri, null);
return returnUri;
} 捕获 (异常 ex) {
Log.e("Insert", ex.toString());
db.关闭();
} finally {
db.关闭();
}
return null;
}

@Override
public int 删除(Uri uri, String 选择条件, String[] 选择参数) {
    final SQLiteDatabase db = DBHelper.获取可写数据库();
    final int 匹配结果 = sUriMatcher.匹配(uri);
    int 删除的行数;
    if (null == 选择条件) 选择条件 = "1";
    try {
        switch (match) {
            case USER: {
删除的行数 = db.删除(
                DBContract.UserEntry.表名, 选择条件, 选择参数);
            break;
        }
        default:
            throw new UnsupportedOperationException("未知的Uri: " + uri);
    }
    if (删除的行数 != 0) {
mContext.获取内容解析器().通知更改(uri, null);
    }
    return 删除的行数;
} 捕获 (异常 ex) {
Log.e("Insert", ex.toString());
} finally {
db.关闭();
}
return 0;
}

```

```

    } finally {
mDBHelper.close();
}
return null;
}

@Override
public Uri insert(Uri uri, ContentValues values) {
    final SQLiteDatabase db = mDBHelper.getWritableDatabase();
    final int match = sUriMatcher.match(uri);
Uri returnUri;
try {

    switch (match) {
        case USER: {
            long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
            if (_id > 0)
returnUri = DBContract.UserEntry.buildUri(_id);
            else
                throw new android.database.SQLException("Error at inserting row in " +
uri);
            break;
        }
        default:
            throw new UnsupportedOperationException("未知的Uri: " + uri);
    }
}
mContext.getContentResolver().notifyChange(uri, null);
return returnUri;
} catch (Exception ex) {
Log.e("Insert", ex.toString());
db.close();
} finally {
db.close();
}
return null;
}

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    final SQLiteDatabase db = DBHelper.getWritableDatabase();
    final int match = sUriMatcher.match(uri);
    int deletedRows;
    if (null == selection) selection = "1";
    try {
        switch (match) {
            case USER: {
deletedRows = db.delete(
                DBContract.UserEntry.TABLE_NAME, selection, selectionArgs);
            break;
        }
        default:
            throw new UnsupportedOperationException("未知的Uri: " + uri);
    }
    if (deletedRows != 0) {
mContext.getContentResolver().notifyChange(uri, null);
    }
    return deletedRows;
} catch (Exception ex) {
Log.e("Insert", ex.toString());
} finally {
db.close();
}
return 0;
}

```

```

}

@Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int updatedRows;
        try {
            switch (match) {
                case USER:
                    updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values, selection,
                        selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("未知的Uri: " + uri);
            }
            if (updatedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return updatedRows;
        } catch (Exception ex) {
            Log.e("Update", ex.toString());
        } finally {
            db.close();
        }
        return -1;
    }
}

```

```

}

@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
    final SQLiteDatabase db = mDBHelper.getWritableDatabase();
    final int match = sUriMatcher.match(uri);
    int updatedRows;
    try {
        switch (match) {
            case USER:
                updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values, selection,
                    selectionArgs);
                break;
            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
        if (updatedRows != 0) {
            mContext.getContentResolver().notifyChange(uri, null);
        }
        return updatedRows;
    } catch (Exception ex) {
        Log.e("Update", ex.toString());
    } finally {
        db.close();
    }
    return -1;
}
}

```

使用方法：

```

public void InsertUser() {
    try {
        ContentValues userValues = getUserData("Jhon", "XXXXX");
        DBProvider dbProvider = new DBProvider(mContext);
        dbProvider.insert(UserEntry.CONTENT_URI, userValues);

    } catch (Exception ex) {
        Log.e("Insert", ex.toString());
    }
}

public ContentValues getUserData(String name, String pass) {
    ContentValues userValues = new ContentValues();
    userValues.put(UserEntry.COLUMN_Name, name);
    userValues.put(UserEntry.COLUMN_Password, pass);
    return userValues;
}

```

第227.7节：从表中删除行

删除表中的所有行

```

//获取可写数据库
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();

```

How to Use:

```

public void InsertUser() {
    try {
        ContentValues userValues = getUserData("Jhon", "XXXXX");
        DBProvider dbProvider = new DBProvider(mContext);
        dbProvider.insert(UserEntry.CONTENT_URI, userValues);

    } catch (Exception ex) {
        Log.e("Insert", ex.toString());
    }
}

public ContentValues getUserData(String name, String pass) {
    ContentValues userValues = new ContentValues();
    userValues.put(UserEntry.COLUMN_Name, name);
    userValues.put(UserEntry.COLUMN_Password, pass);
    return userValues;
}

```

Section 227.7: Delete row(s) from the table

To delete all rows from the table

```

//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();

```

删除表中所有行并返回删除的行数

```
//获取可写数据库
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

删除带有WHERE条件的行

```
//获取可写数据库
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//对于多个条件，用AND连接它们
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

第227.8节：更新表中的一行

```
// 你需要一个可写的数据库来更新一行
final SQLiteDatabase database = openHelper.getWritableDatabase();

// 创建一个ContentValues实例，包含每列的最新数据
// 与插入数据不同，你还需要指定主键列的值
final ContentValues values = new ContentValues();
values.put(COLUMN_ID, model.getId());
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// 该调用执行更新操作
// 返回值告诉你更新了多少行。
final int count = database.update(
    TABLE_NAME,           // 要更新数据的表名      values,           // 包含新数据的ContentValues实例
    COLUMN_ID + " = ?", // 指定更新哪一行的选择条件。?符号是参数占位符。
    new String[] {         // 选择条件的实际参数，作为String数组。
        String.valueOf(model.getId())
    }
);
```

第227.9节：执行事务

事务可用于对数据库进行多项更改，且这些更改是原子性的。任何普通事务遵循以下模式：

```
// 你需要一个可写的数据库来执行事务
final SQLiteDatabase database = openHelper.getWritableDatabase();

// 此调用启动一个事务
database.beginTransaction();
```

To delete all rows from the table and get the count of the deleted row in return value

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

To delete row(s) with WHERE condition

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//for multiple condition, join them with AND
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

Section 227.8: Updating a row in a table

```
// You need a writable database to update a row
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the up to date data for each column
// Unlike when inserting data you need to specify the value for the PRIMARY KEY column as well
final ContentValues values = new ContentValues();
values.put(COLUMN_ID, model.getId());
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value tells you how many rows have been updated.
final int count = database.update(
    TABLE_NAME,           // The table name in which the data will be updated
    values,               // The ContentValues instance with the new data
    COLUMN_ID + " = ?", // The selection which specifies which row is updated. ? symbols are
    parameters.          // The actual parameters for the selection as a String[].
    new String[] {         String.valueOf(model.getId())
    }
);
```

Section 227.9: Performing a Transaction

Transactions can be used to make multiple changes to the database atomically. Any normal transaction follows this pattern:

```
// You need a writable database to perform transactions
final SQLiteDatabase database = openHelper.getWritableDatabase();

// This call starts a transaction
database.beginTransaction();
```

// 使用try/finally是确保即使发生异常或其他问题也能可靠结束事务的关键。

```
try {  
  
    // 在这里你可以对数据库进行修改  
    database.insert(TABLE_CARS, null, productValues);  
    database.update(TABLE_BUILDINGS, buildingValues, COLUMN_ID + " = ?", new String[] {  
        String.valueOf(buildingId) });  
  
    // 此调用将事务标记为成功。  
    // 这会导致在事务结束时将更改写入数据库。  
    database.setTransactionSuccessful();  
} finally {  
    // 该调用结束一个事务。  
    // 如果未调用 setTransactionSuccessful(), 则所有更改  
    // 将被回滚, 数据库不会被修改。  
    database.endTransaction();  
}
```

在一个活动事务中调用 beginTransaction() 不会产生任何效果。

第227.10节：从assets文件夹创建数据库

将你的 dbname.sqlite 或 dbname.db 文件放入项目的 assets 文件夹中。

```
public class Databasehelper extends SQLiteOpenHelper {  
    public static final String TAG = Databasehelper.class.getSimpleName();  
    public static int flag;  
    // 你放入 assets 文件夹中的数据库文件的准确名称及其扩展名。  
    static String DB_NAME = "dbname.sqlite";  
    private final Context myContext;  
    String outFileName = "";  
    private String DB_PATH;  
    private SQLiteDatabase db;  
  
    public Databasehelper(Context context) {  
        super(context, DB_NAME, null, 1);  
        this.myContext = context;  
        ContextWrapper cw = new ContextWrapper(context);  
        DB_PATH = cw.getFilesDir().getAbsolutePath() + "/databases/";  
        Log.e(TAG, "Databasehelper: DB_PATH " + DB_PATH);  
        outFileName = DB_PATH + DB_NAME;  
        File file = new File(DB_PATH);  
        Log.e(TAG, "Databasehelper: " + file.exists());  
        if (!file.exists()) {  
            file.mkdir();  
        }  
    }  
  
    /**  
     * 在系统上创建一个空数据库，并用您自己的数据库重写它。  
     */  
    public void createDataBase() throws IOException {  
        boolean dbExist = checkDataBase();  
        if (dbExist) {  
            //什么也不做 - 数据库已存在  
        } else {  
            //调用此方法将在默认系统路径中创建一个空数据库  
            path  
            //你的应用程序，因此我们将能够用我们的数据库覆盖该数据库。  
        }  
    }
```

// Using try/finally is essential to reliably end transactions even
// if exceptions or other problems occur.

```
try {  
  
    // Here you can make modifications to the database  
    database.insert(TABLE_CARS, null, productValues);  
    database.update(TABLE_BUILDINGS, buildingValues, COLUMN_ID + " = ?", new String[] {  
        String.valueOf(buildingId) });  
  
    // This call marks a transaction as successful.  
    // This causes the changes to be written to the database once the transaction ends.  
    database.setTransactionSuccessful();  
} finally {  
    // This call ends a transaction.  
    // If setTransactionSuccessful() has not been called then all changes  
    // will be rolled back and the database will not be modified.  
    database.endTransaction();  
}
```

Calling beginTransaction() inside of an active transactions has no effect.

Section 227.10: Create Database from assets folder

Put your dbname.sqlite or dbname.db file in assets folder of your project.

```
public class Databasehelper extends SQLiteOpenHelper {  
    public static final String TAG = Databasehelper.class.getSimpleName();  
    public static int flag;  
    // Exact Name of your db file that you put in assets folder with extension.  
    static String DB_NAME = "dbname.sqlite";  
    private final Context myContext;  
    String outFileName = "";  
    private String DB_PATH;  
    private SQLiteDatabase db;  
  
    public Databasehelper(Context context) {  
        super(context, DB_NAME, null, 1);  
        this.myContext = context;  
        ContextWrapper cw = new ContextWrapper(context);  
        DB_PATH = cw.getFilesDir().getAbsolutePath() + "/databases/";  
        Log.e(TAG, "Databasehelper: DB_PATH " + DB_PATH);  
        outFileName = DB_PATH + DB_NAME;  
        File file = new File(DB_PATH);  
        Log.e(TAG, "Databasehelper: " + file.exists());  
        if (!file.exists()) {  
            file.mkdir();  
        }  
    }  
  
    /**  
     * Creates a empty database on the system and rewrites it with your own database.  
     */  
    public void createDataBase() throws IOException {  
        boolean dbExist = checkDataBase();  
        if (dbExist) {  
            //do nothing - database already exist  
        } else {  
            //By calling this method and empty database will be created into the default system  
            path  
            //of your application so we are gonna be able to overwrite that database with our  
            database.  
        }  
    }
```

```

        this.getReadableDatabase();
        try {
copyDataBase();
    } catch (IOException e) {
        throw new Error("Error copying database");
    }
}

/**
* 检查数据库是否已存在，以避免每次打开
* 应用程序时都重新复制文件。
*
* @return 如果存在返回true，不存在返回false
*/
private boolean checkDataBase() {
    SQLiteDatabase checkDB = null;
    try {
checkDB = SQLiteDatabase.openDatabase(outFileName, null,
SQLiteDatabase.OPEN_READWRITE);
    } catch (SQLiteException e) {
        try {
copyDataBase();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}

    if (checkDB != null) {
        checkDB.close();
    }
    return checkDB != null ? true : false;
}

/**
* 将您的数据库从本地资源文件夹复制到刚创建的空数据库中，
*****
* 从系统文件夹中可以访问和操作该数据库。
* 这是通过传输字节流完成的。
*/
private void copyDataBase() throws IOException {
Log.i("Database",
        "新的数据库正在复制到设备！");
byte[] buffer = new byte[1024];
OutputStream myOutput = null;
int length;
// 打开本地数据库作为输入流
InputStream myInput = null;
try {
myInput = myContext.getAssets().open(DB_NAME);
    // 将字节从输入文件传输到
    // 输出文件
myOutput = new FileOutputStream(DB_PATH + DB_NAME);
    while ((length = myInput.read(buffer)) > 0) {
        myOutput.write(buffer, 0, length);
    }
myOutput.close();
    myOutput.flush();
    myInput.close();
    Log.i("Database",

```

```

        this.getReadableDatabase();
        try {
copyDataBase();
    } catch (IOException e) {
        throw new Error("Error copying database");
    }
}

/**
* Check if the database already exist to avoid re-copying the file each time you open the
application.
*
* @return true if it exists, false if it doesn't
*/
private boolean checkDataBase() {
    SQLiteDatabase checkDB = null;
    try {
checkDB = SQLiteDatabase.openDatabase(outFileName, null,
SQLiteDatabase.OPEN_READWRITE);
    } catch (SQLiteException e) {
        try {
copyDataBase();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}

    if (checkDB != null) {
        checkDB.close();
    }
    return checkDB != null ? true : false;
}

/**
* Copies your database from your local assets-folder to the just created empty database in
the
* system folder, from where it can be accessed and handled.
* This is done by transferring bytestream.
*/
private void copyDataBase() throws IOException {
Log.i("Database",
        "New database is being copied to device!");
byte[] buffer = new byte[1024];
OutputStream myOutput = null;
int length;
// Open your local db as the input stream
InputStream myInput = null;
try {
    myInput = myContext.getAssets().open(DB_NAME);
    // transfer bytes from the inputfile to the
    // outputfile
    myOutput = new FileOutputStream(DB_PATH + DB_NAME);
    while ((length = myInput.read(buffer)) > 0) {
        myOutput.write(buffer, 0, length);
    }
    myOutput.close();
    myOutput.flush();
    myInput.close();
    Log.i("Database",

```

```

        "新数据库已复制到设备！");
    } catch (IOException e) {
e.printStackTrace();
    }

    public void openDataBase() throws SQLException {
        //打开数据库
        String myPath = DB_PATH + DB_NAME;
db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE);
        Log.e(TAG, "openDataBase: Open " + db.isOpen());
    }

    @Override
    public synchronized void close() {
        if (db != null)
db.close();
        super.close();
    }

    public void onCreate(SQLiteDatabase arg0) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
    }
}

```

这是你如何在活动中访问数据库对象的方法。

```
// 在你的活动中创建 Databasehelper 类对象。
private Databasehelper db;
```

然后在 onCreate 方法中初始化它，并调用 createDatabase() 方法，如下所示。

```

db = new Databasehelper(MainActivity.this);
try {
db.createDataBase();
} catch (Exception e) {
e.printStackTrace();
}

```

按照以下示例执行所有的插入、更新、删除和查询操作。

```

String query = "select Max(Id) as Id from " + TABLE_NAME;
db.openDataBase();
int count = db.getId(query);
db.close();

```

第227.11节：将图像存储到SQLite

设置数据库

```
public class DatabaseHelper extends SQLiteOpenHelper {
    // 数据库版本
    private static final int DATABASE_VERSION = 1;
```

```

        "New database has been copied to device!");
    } catch (IOException e) {
        e.printStackTrace();
    }

    public void openDataBase() throws SQLException {
        //Open the database
        String myPath = DB_PATH + DB_NAME;
db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE);
        Log.e(TAG, "openDataBase: Open " + db.isOpen());
    }

    @Override
    public synchronized void close() {
        if (db != null)
            db.close();
        super.close();
    }

    public void onCreate(SQLiteDatabase arg0) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
    }
}

```

Here is How you can access database object to your activity.

```
// Create Databasehelper class object in your activity.
private Databasehelper db;
```

Then in onCreate Method initialize it and call createDatabase() Method as show below.

```

db = new Databasehelper(MainActivity.this);
try {
    db.createDataBase();
} catch (Exception e) {
    e.printStackTrace();
}

```

Perform all of your insert, update, delete and select operation as shown below.

```

String query = "select Max(Id) as Id from " + TABLE_NAME;
db.openDataBase();
int count = db.getId(query);
db.close();

```

Section 227.11: Store image into SQLite

Setting Up the database

```
public class DatabaseHelper extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;
```

```

// 数据库名称
private static final String DATABASE_NAME = "database_name";

// 表名
private static final String DB_TABLE = "table_image";

// 列名
private static final String KEY_NAME = "image_name";
private static final String KEY_IMAGE = "image_data";

// 表创建语句
private static final String CREATE_TABLE_IMAGE = "CREATE TABLE " + DB_TABLE + "(" +
    KEY_NAME + " TEXT," +
    KEY_IMAGE + " BLOB);";

public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {

    // 创建表
    db.execSQL(CREATE_TABLE_IMAGE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // 升级时删除旧表
    db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);

    // 创建新表
    onCreate(db);
}

```

插入数据库：

```

public void addEntry( String name, byte[] image) throws SQLiteException{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(KEY_NAME, name);
    cv.put(KEY_IMAGE, image);
    database.insert( DB_TABLE, null, cv );
}

```

检索数据：

```
byte[] image = cursor.getBlob(1);
```

注意：

- 在插入数据库之前，您需要先将Bitmap图像转换为字节数组，然后通过数据库查询应用它。
- 从数据库检索时，您肯定会得到图像的字节数组，您需要做的是将字节数组转换回原始图像。因此，您必须使用BitmapFactory进行解码。

下面是一个实用类，希望能帮到您：

```

// Database Name
private static final String DATABASE_NAME = "database_name";

// Table Names
private static final String DB_TABLE = "table_image";

// column names
private static final String KEY_NAME = "image_name";
private static final String KEY_IMAGE = "image_data";

// Table create statement
private static final String CREATE_TABLE_IMAGE = "CREATE TABLE " + DB_TABLE + "(" +
    KEY_NAME + " TEXT," +
    KEY_IMAGE + " BLOB);";

public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {

    // creating table
    db.execSQL(CREATE_TABLE_IMAGE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // on upgrade drop older tables
    db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);

    // create new table
    onCreate(db);
}

```

Insert in the Database:

```

public void addEntry( String name, byte[] image) throws SQLiteException{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(KEY_NAME, name);
    cv.put(KEY_IMAGE, image);
    database.insert( DB_TABLE, null, cv );
}

```

Retrieving data:

```
byte[] image = cursor.getBlob(1);
```

Note:

- Before inserting into database, you need to convert your Bitmap image into byte array first then apply it using database query.
- When retrieving from database, you certainly have a byte array of image, what you need to do is to convert byte array back to original image. So, you have to make use of BitmapFactory to decode.

Below is an Utility class which I hope could help you:

```

public class DbBitmapUtility {

    // 将位图转换为字节数组
    public static byte[] getBytes(Bitmap bitmap) {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap.compress(CompressFormat.PNG, 0, stream);
        return stream.toByteArray();
    }

    // 将字节数组转换为位图
    public static Bitmap getImage(byte[] image) {
        return BitmapFactory.decodeByteArray(image, 0, image.length);
    }
}

```

第227.12节：导出和导入数据库

例如，您可能想导入和导出数据库以备份。别忘了权限设置。

```

public void exportDatabase(){
    try {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//MY.PACKAGE.NAME//databases//MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File currentDB = new File(data, currentDBPath);
        File backupDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
        src.close();
        dst.close();

        Toast.makeText(c, c.getResources().getString(R.string.exporterentToast),
        Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
        Toast.LENGTH_SHORT).show();
        Log.d("Main", e.toString());
    }
}

```

```

public void importDatabase(){
    try {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//" + "MY.PACKAGE.NAME" + "//databases//" +
        "MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File backupDB = new File(data, currentDBPath);
        File currentDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
    }
}

```

```

public class DbBitmapUtility {

    // convert from bitmap to byte array
    public static byte[] getBytes(Bitmap bitmap) {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap.compress(CompressFormat.PNG, 0, stream);
        return stream.toByteArray();
    }

    // convert from byte array to bitmap
    public static Bitmap getImage(byte[] image) {
        return BitmapFactory.decodeByteArray(image, 0, image.length);
    }
}

```

Section 227.12: Exporting and importing a database

You might want to import and export your database for backups for example. Don't forget about the permissions.

```

public void exportDatabase(){
    try {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//MY.PACKAGE.NAME//databases//MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File currentDB = new File(data, currentDBPath);
        File backupDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
        src.close();
        dst.close();

        Toast.makeText(c, c.getResources().getString(R.string.exporterentToast),
        Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
        Toast.LENGTH_SHORT).show();
        Log.d("Main", e.toString());
    }
}

```

```

public void importDatabase(){
    try {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//" + "MY.PACKAGE.NAME" + "//databases//" +
        "MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File backupDB = new File(data, currentDBPath);
        File currentDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
    }
}

```

```
src.close();
dst.close();
Toast.makeText(c, c.getResources().getString(R.string.importerentToast),
Toast.LENGTH_LONG).show();
}
catch (Exception e) {
Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
}
}
```

```
src.close();
dst.close();
Toast.makeText(c, c.getResources().getString(R.string.importerentToast),
Toast.LENGTH_LONG).show();
}
catch (Exception e) {
Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
}
}
```

第228章：使用ContentValues类访问SQLite数据库

第228.1节：在SQLite数据库中插入和更新行

首先，您需要打开您的SQLite数据库，方法如下：

```
SQLiteDatabase myDataBase;  
String mPath = dbhelper.DATABASE_PATH + dbhelper.DATABASE_NAME;  
myDataBase = SQLiteDatabase.openDatabase(mPath, null, SQLiteDatabase.OPEN_READWRITE);
```

打开数据库后，您可以使用ContentValues类轻松插入或更新行。以下示例假设名字由str_edtfname给出，姓氏由str_edtlnname给出。您还需要将table_name替换为您想要修改的表名。

插入数据

```
ContentValues values = new ContentValues();  
values.put("First_Name", str_edtfname);  
values.put("Last_Name", str_edtlnname);  
myDataBase.insert("table_name", null, values);
```

更新数据

```
ContentValues values = new ContentValues();  
values.put("First_Name", str_edtfname);  
values.put("Last_Name", str_edtlnname);  
myDataBase.update("table_name", values, "id" + " = ?", new String[] {id});
```

Chapter 228: Accessing SQLite databases using the ContentValues class

Section 228.1: Inserting and updating rows in a SQLite database

First, you need to open your SQLite database, which can be done as follows:

```
SQLiteDatabase myDataBase;  
String mPath = dbhelper.DATABASE_PATH + dbhelper.DATABASE_NAME;  
myDataBase = SQLiteDatabase.openDatabase(mPath, null, SQLiteDatabase.OPEN_READWRITE);
```

After opening the database, you can easily insert or update rows by using the [ContentValues](#) class. The following examples assume that a first name is given by str_edtfname and a last name by str_edtlnname. You also need to replace table_name by the name of your table that you want to modify.

Inserting data

```
ContentValues values = new ContentValues();  
values.put("First_Name", str_edtfname);  
values.put("Last_Name", str_edtlnname);  
myDataBase.insert("table_name", null, values);
```

Updating data

```
ContentValues values = new ContentValues();  
values.put("First_Name", str_edtfname);  
values.put("Last_Name", str_edtlnname);  
myDataBase.update("table_name", values, "id" + " = ?", new String[] {id});
```

第229章：Firebase

[Firebase](#) 是一个移动和网页应用平台，提供工具和基础设施，旨在帮助开发者构建高质量的应用程序。

功能

Firebase 云消息传递、Firebase 认证、实时数据库、Firebase 存储、Firebase 托管、Firebase 测试实验室（针对 Android）、Firebase 崩溃报告。

第229.1节：将 Firebase 添加到您的 Android 项目

以下是创建 Firebase 项目并将其

与 Android 应用连接所需的简化步骤（基于官方文档）。

将 Firebase 添加到您的应用

1. 在 Firebase 控制台中创建一个 [Firebase](#) 项目，然后点击创建新项目。
2. 点击将 Firebase 添加到您的 Android 应用并按照设置步骤操作。
3. 提示时，输入您的应用包名。
请输入您的应用正在使用的完整包名；此项只能在您将应用添加到 Firebase 项目时设置。
4. 最后，您将下载一个google-services.json文件。您可以随时重新下载此文件。
5. 如果尚未操作，请将google-services.json文件复制到项目的模块文件夹中，
通常是app/。

下一步是添加 SDK，将 Firebase 库集成到项目中。

添加 SDK

要将 Firebase 库集成到您自己的项目中，您需要执行一些基本任务来准备您的 Android Studio 项目。您可能已经在将 Firebase 添加到您的应用时完成了这些步骤。

1. 在你的根目录级别的build.gradle文件中添加规则，以包含google-services插件：

```
buildscript {  
    // ...  
dependencies {  
    // ...  
classpath 'com.google.gms:google-services:3.1.0'  
}  
}
```

然后，在你的模块Gradle文件（通常是app/build.gradle）中，在文件底部添加apply plugin行以启用Gradle插件：

```
apply plugin: 'com.android.application'  
  
android {  
    // ...  
}
```

Chapter 229: Firebase

[Firebase](#) is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps.

Features

Firebase Cloud Messaging, Firebase Auth, Realtime Database, Firebase Storage, Firebase Hosting, Firebase Test Lab for Android, Firebase Crash Reporting.

Section 229.1: Add Firebase to Your Android Project

Here are simplified steps (based on the [official documentation](#)) required to create a Firebase project and connect it with an Android app.

Add Firebase to your app

1. Create a Firebase project in the [Firebase console](#) and click **Create New Project**.
2. Click **Add Firebase to your Android app** and follow the setup steps.
3. When prompted, enter your **app's package name**.
It's important to enter the fully qualified package name your app is using; this can only be set when you add an app to your Firebase project.
4. At the end, you'll download a `google-services.json` file. You can download this file again at any time.
5. If you haven't done so already, copy the `google-services.json` file into your project's module folder, typically `app/`.

The next step is to Add the SDK to integrate the Firebase libraries in the project.

Add the SDK

To integrate the Firebase libraries into one of your own projects, you need to perform a few basic tasks to prepare your Android Studio project. You may have already done this as part of adding Firebase to your app.

1. Add rules to your root-level `build.gradle` file, to include the **google-services plugin**:

```
buildscript {  
    // ...  
dependencies {  
    // ...  
classpath 'com.google.gms:google-services:3.1.0'  
}  
}
```

Then, in your module Gradle file (usually the `app/build.gradle`), add the apply plugin line at the bottom of the file to enable the Gradle plugin:

```
apply plugin: 'com.android.application'  
  
android {  
    // ...  
}
```

```

dependencies {
    // ...
    compile 'com.google.firebaseio:firebase-core:11.0.4'
}

// 在底部添加此行
apply plugin: 'com.google.gms.google-services'

```

最后一步是使用一个或多个可用库为不同的Firebase功能添加Firebase SDK的依赖项。

| Gradle 依赖项 | 服务 |
|---|-----------|
| com.google.firebaseio:firebase-core:11.0.4 | 分析 |
| com.google.firebaseio:firebase-database:11.0.4 | 实时数据库 |
| com.google.firebaseio:firebase-storage:11.0.4 | 存储 |
| com.google.firebaseio:firebase-crash:11.0.4 | 崩溃报告 |
| com.google.firebaseio:firebase-auth:11.0.4 | 身份验证 |
| com.google.firebaseio:firebase-messaging:11.0.4 | 云消息 / 通知 |
| com.google.firebaseio:firebase-config:11.0.4 | 远程配置 |
| com.google.firebaseio:firebase-invites:11.0.4 | 邀请 / 动态链接 |
| com.google.firebaseio:firebase-ads:11.0.4 | AdMob |
| com.google.android.gms:play-services-appindexing:11.0.4 | 应用索引 |

```

dependencies {
    // ...
    compile 'com.google.firebaseio:firebase-core:11.0.4'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'

```

The final step is to add the dependencies for the Firebase SDK using one or more **libraries available** for the different Firebase features.

| Gradle Dependency Line | Service |
|---|---------------------------------|
| com.google.firebaseio:firebase-core:11.0.4 | Analytics |
| com.google.firebaseio:firebase-database:11.0.4 | Realtime Database |
| com.google.firebaseio:firebase-storage:11.0.4 | Storage |
| com.google.firebaseio:firebase-crash:11.0.4 | Crash Reporting |
| com.google.firebaseio:firebase-auth:11.0.4 | Authentication |
| com.google.firebaseio:firebase-messaging:11.0.4 | Cloud Messaging / Notifications |
| com.google.firebaseio:firebase-config:11.0.4 | Remote Config |
| com.google.firebaseio:firebase-invites:11.0.4 | Invites / Dynamic Links |
| com.google.firebaseio:firebase-ads:11.0.4 | AdMob |
| com.google.android.gms:play-services-appindexing:11.0.4 | App Indexing |

第229.2节：更新Firebase用户的电子邮件

```

public class ChangeEmailActivity extends AppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseAuth mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "请输入邮箱");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "请输入有效的邮箱");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mAuth.getCurrentUser();
    }
}

```

Section 229.2: Updating a Firebase users's email

```

public class ChangeEmailActivity extends AppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseAuth mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter valid email");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mAuth.getCurrentUser();
    }
}

```

```

private void changeEmail(String email) {
    DialogUtils.showProgressDialog(this, "正在更改邮箱", "请稍候...", false);
    mFirebaseUser.updateEmail(email)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
                showToast("邮箱更新成功。");
                return;
            }

            if (task.getException() instanceof
                FirebaseAuthRecentLoginRequiredException) {
                FragmentManager fm = getSupportFragmentManager();
                ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
                    ReAuthenticateDialogFragment();
                reAuthenticateDialogFragment.show(fm,
                    reAuthenticateDialogFragment.getClass().getSimpleName());
            }
        });
    });

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}

```

第229.3节：创建Firebase用户

```

public class SignUpActivity extends AppCompatActivity {

    @BindView(R.id.tIETSignUpEmail)
    EditText mEditTextEmail;
    @BindView(R.id.tIETSignUpPassword)
    EditText mEditTextPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @OnClick(R.id.btnSignUpSignUp)
    void signUp() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            mEditTextEmail.setError("请输入邮箱");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {

```

```

private void changeEmail(String email) {
    DialogUtils.showProgressDialog(this, "Changing Email", "Please wait...", false);
    mFirebaseUser.updateEmail(email)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
                showToast("Email updated successfully.");
                return;
            }

            if (task.getException() instanceof
                FirebaseAuthRecentLoginRequiredException) {
                FragmentManager fm = getSupportFragmentManager();
                ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
                    ReAuthenticateDialogFragment();
                reAuthenticateDialogFragment.show(fm,
                    reAuthenticateDialogFragment.getClass().getSimpleName());
            }
        });
    });

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}

```

Section 229.3: Create a Firebase user

```

public class SignUpActivity extends AppCompatActivity {

    @BindView(R.id.tIETSignUpEmail)
    EditText mEditTextEmail;
    @BindView(R.id.tIETSignUpPassword)
    EditText mEditTextPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @OnClick(R.id.btnSignUpSignUp)
    void signUp() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            mEditTextEmail.setError("Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {

```

```

mEditEmail.setError("请输入有效的邮箱");
    return;
}

if (TextUtils.isEmpty(mEditPassword.getText())) {
    mEditPassword.setError("请输入密码");
    return;
}

createUserWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
}

private void createUserWithEmailAndPassword(String email, String password) {
    DialogUtils.showProgressDialog(this, "", getString(R.string.str_creating_account), false);
    mFirebaseAuth
.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (!task.isSuccessful()) {
                Toast.makeText(SignUpActivity.this, task.getException().getMessage(),
                    Toast.LENGTH_SHORT).show();
                DialogUtils.dismissProgressDialog();
            } else {
                Toast.makeText(SignUpActivity.this,
R.string.str_registration_successful, Toast.LENGTH_SHORT).show();
                DialogUtils.dismissProgressDialog();
                startActivity(new Intent(SignUpActivity.this, HomeActivity.class));
            }
        }
    });
}

@Override
protected int getLayoutResourceId() {
    return R.layout.activity_sign_up;
}
}

```

第229.4节：修改密码

```

public class ChangePasswordActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {
    @BindView(R.id.et_change_password)
    EditText mEditText;
    private FirebaseAuth mFirebaseUser;

    @OnClick(R.id.btn_change_password)
    void onChangePasswordClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "请输入密码");
            return;
        }

        changePassword(mEditText.getText().toString());
    }
}

```

```

mEditEmail.setError("Please enter valid email");
    return;
}

if (TextUtils.isEmpty(mEditPassword.getText())) {
    mEditPassword.setError("Please enter password");
    return;
}

createUserWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
}

private void createUserWithEmailAndPassword(String email, String password) {
    DialogUtils.showProgressDialog(this, "", getString(R.string.str_creating_account), false);
    mFirebaseAuth
.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (!task.isSuccessful()) {
                Toast.makeText(SignUpActivity.this, task.getException().getMessage(),
                    Toast.LENGTH_SHORT).show();
                DialogUtils.dismissProgressDialog();
            } else {
                Toast.makeText(SignUpActivity.this,
R.string.str_registration_successful, Toast.LENGTH_SHORT).show();
                DialogUtils.dismissProgressDialog();
                startActivity(new Intent(SignUpActivity.this, HomeActivity.class));
            }
        }
    });
}

@Override
protected int getLayoutResourceId() {
    return R.layout.activity_sign_up;
}
}

```

Section 229.4: Change Password

```

public class ChangePasswordActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {
    @BindView(R.id.et_change_password)
    EditText mEditText;
    private FirebaseAuth mFirebaseUser;

    @OnClick(R.id.btn_change_password)
    void onChangePasswordClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter password");
            return;
        }

        changePassword(mEditText.getText().toString());
    }
}

```

```

private void changePassword(String password) {
    DialogUtils.showProgressDialog(this, "正在更改密码", "请稍候...", false);
    mFirebaseUser.updatePassword(password)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
                showToast("密码更新成功。");
                return;
            }

            if (task.getException() instanceof
                FirebaseAuthRecentLoginRequiredException) {
                FragmentManager fm = getSupportFragmentManager();
                ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
                ReAuthenticateDialogFragment();
                reAuthenticateDialogFragment.show(fm,
                    reAuthenticateDialogFragment.getClass().getSimpleName());
            }
        });
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mAuth.getCurrentUser();
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_password;
    }

    @Override
    public void onReauthenticateSuccess() {
        changePassword(mEditText.getText().toString());
    }
}

```

```

private void changePassword(String password) {
    DialogUtils.showProgressDialog(this, "Changing Password", "Please wait...", false);
    mFirebaseUser.updatePassword(password)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
                showToast("Password updated successfully.");
                return;
            }

            if (task.getException() instanceof
                FirebaseAuthRecentLoginRequiredException) {
                FragmentManager fm = getSupportFragmentManager();
                ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
                ReAuthenticateDialogFragment();
                reAuthenticateDialogFragment.show(fm,
                    reAuthenticateDialogFragment.getClass().getSimpleName());
            }
        }
    });

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mAuth.getCurrentUser();
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_password;
    }

    @Override
    public void onReauthenticateSuccess() {
        changePassword(mEditText.getText().toString());
    }
}

```

第229.5节：Firebase云消息传递

首先，您需要按照本主题中描述的步骤，将Firebase添加到您的Android项目中以设置项目。

设置Firebase和FCM SDK

将FCM依赖项添加到您的应用级build.gradle文件中

```

dependencies {
    compile 'com.google.firebaseio:firebase-messaging:11.0.4'
}

```

并且在最底部（这很重要）添加：

```

// 在底部添加此行
apply plugin: 'com.google.gms.google-services'

```

Section 229.5: Firebase Cloud Messaging

First of all you need to setup your project adding Firebase to your Android project following the steps described in this topic.

Set up Firebase and the FCM SDK

Add the FCM dependency to your app-level build.gradle file

```

dependencies {
    compile 'com.google.firebaseio:firebase-messaging:11.0.4'
}

```

And at the very bottom (this is important) add:

```

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'

```

编辑您的应用清单

将以下内容添加到您的应用清单中：

- 一个继承自FirebaseMessagingService的服务。如果您想在应用后台接收通知之外进行任何消息处理，这是必需的。
- 一个继承自FirebaseInstanceIdService的服务，用于处理注册令牌的创建、轮换和更新。

例如：

```
<service
    android:name=".MyInstanceIdListenerService">
    <intent-filter>
        <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>
<service
    android:name=".MyFcmListenerService">
    <intent-filter>
        <action android:name="com.google.firebaseio.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

这里是两个服务的简单实现。

要获取当前的注册令牌，请继承FirebaseInstanceIdService类并重写onTokenRefresh()方法：

```
public class MyInstanceIdListenerService extends FirebaseInstanceIdService {

    // 如果InstanceId令牌被更新时调用。如果之前的令牌安全性被
    // 破坏，则会发生此情况。此调用由InstanceId提供者发起。
    @Override
    public void onTokenRefresh() {
        // 获取更新后的InstanceId令牌。
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();

        // 将此令牌发送到您的服务器或本地存储
    }
}
```

要接收消息，请使用继承FirebaseMessagingService的服务并重写onMessageReceived方法。

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * 当收到消息时调用。
     *
     * @param remoteMessage 表示从 Firebase 云消息传递接收到的消息的对象。
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // 检查消息是否包含数据负载。
        if (remoteMessage.getData().size() > 0) {
```

Edit your app manifest

Add the following to your app's manifest:

- A service that extends FirebaseMessagingService. This is required if you want to do any message handling beyond receiving notifications on apps in the background.
- A service that extends FirebaseInstanceIdService to handle the creation, rotation, and updating of registration tokens.

For example:

```
<service
    android:name=".MyInstanceIdListenerService">
    <intent-filter>
        <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>
<service
    android:name=".MyFcmListenerService">
    <intent-filter>
        <action android:name="com.google.firebaseio.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Here are simple implementations of the 2 services.

To retrieve the current registration token extend the FirebaseInstanceIdService class and override the onTokenRefresh() method:

```
public class MyInstanceIdListenerService extends FirebaseInstanceIdService {

    // Called if InstanceID token is updated. Occurs if the security of the previous token had been
    // compromised. This call is initiated by the InstanceID provider.
    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();

        // Send this token to your server or store it locally
    }
}
```

To receive messages, use a service that extends FirebaseMessagingService and override the onMessageReceived method.

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
```

```

Log.d(TAG, "消息数据负载: " + remoteMessage.getData());
Map<String, String> data = remoteMessage.getData();
}

// 检查消息是否包含通知负载。
if (remoteMessage.getNotification() != null) {
Log.d(TAG, "消息通知内容: " + remoteMessage.getNotification().getBody());
}

// 对此执行你想做的操作，发布你自己的通知，或更新本地状态
}

```

在Firebase中，可以根据用户的行为将用户分组，例如“应用版本、免费用户、付费用户或任何特定规则”，然后通过Fire base中的Topic功能向特定组发送通知。

注册用户到主题使用

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

然后在 Firebase 控制台，通过主题名称发送通知

更多信息请参见专门的主题 Firebase Cloud Messaging。

第229.6节：Firebase 存储操作

通过此示例，您将能够执行以下操作：

1. 连接到 Firebase 存储
2. 创建一个名为“images”的目录
3. 在 images 目录中上传文件
4. 从 images 目录下载文件
5. 从 images 目录删除文件

```

public class MainActivity extends AppCompatActivity {

private static final int REQUEST_CODE_PICK_IMAGE = 1;
private static final int PERMISSION_READ_WRITE_EXTERNAL_STORAGE = 2;

private FirebaseStorage mFirebaseStorage;
private StorageReference mStorageReference;
private StorageReference mStorageReferenceImages;
private Uri mUri;
private ImageView mImageView;
private ProgressDialog mProgressDialog;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    mImageView = (ImageView) findViewById(R.id.imageView);
    setSupportActionBar(toolbar);

    // 创建 Firebase Storage 实例
    mFirebaseStorage = FirebaseStorage.getInstance();
}

private void pickImage() {
Intent intent = new Intent(Intent.ACTION_PICK,

```

```

Log.d(TAG, "Message data payload: " + remoteMessage.getData());
Map<String, String> data = remoteMessage.getData();
}

// Check if message contains a notification payload.
if (remoteMessage.getNotification() != null) {
    Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
}

// do whatever you want with this, post your own notification, or update local state
}

```

在 Firebase 中可以按行为将用户分组，如“AppVersion, free user, purchase user, or any specific rules”并将其发送给特定组。要实现此功能，请在 Firebase 控制台中注册主题。

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

然后在 Firebase 控制台，通过主题名称发送通知

更多信息请参见专门的主题 Firebase Cloud Messaging。

Section 229.6: Firebase Storage Operations

With this example, you will be able to perform following operations:

1. Connect to Firebase Storage
2. Create a directory named “images”
3. Upload a file in images directory
4. Download a file from images directory
5. Delete a file from images directory

```

public class MainActivity extends AppCompatActivity {

private static final int REQUEST_CODE_PICK_IMAGE = 1;
private static final int PERMISSION_READ_WRITE_EXTERNAL_STORAGE = 2;

private FirebaseStorage mFirebaseStorage;
private StorageReference mStorageReference;
private StorageReference mStorageReferenceImages;
private Uri mUri;
private ImageView mImageView;
private ProgressDialog mProgressDialog;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    mImageView = (ImageView) findViewById(R.id.imageView);
    setSupportActionBar(toolbar);

    // Create an instance of Firebase Storage
    mFirebaseStorage = FirebaseStorage.getInstance();
}

private void pickImage() {
    Intent intent = new Intent(Intent.ACTION_PICK,

```

```

        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        intent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
        startActivityForResult(intent, REQUEST_CODE_PICK_IMAGE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            if (requestCode == REQUEST_CODE_PICK_IMAGE) {
                String filePath = FileUtil.getPath(this, data.getData());
                mUri = Uri.fromFile(new File(filePath));
                uploadFile(mUri);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
    int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_READ_WRITE_EXTERNAL_STORAGE) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                pickImage();
            }
        }
    }

    private void showProgressDialog(String title, String message) {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setMessage(message);
        else
            mProgressDialog = ProgressDialog.show(this, title, message, true, false);
    }

    private void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.dismiss();
    }

    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    public void showHorizontalProgressDialog(String title, String body) {

        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
        } else {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
            mProgressDialog.setIndeterminate(false);
            mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            mProgressDialog.setProgress(0);
            mProgressDialog.setMax(100);
            mProgressDialog.setCancelable(false);
            mProgressDialog.show();
        }
    }
}

```

```

        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        intent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
        startActivityForResult(intent, REQUEST_CODE_PICK_IMAGE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            if (requestCode == REQUEST_CODE_PICK_IMAGE) {
                String filePath = FileUtil.getPath(this, data.getData());
                mUri = Uri.fromFile(new File(filePath));
                uploadFile(mUri);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
    int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_READ_WRITE_EXTERNAL_STORAGE) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                pickImage();
            }
        }
    }

    private void showProgressDialog(String title, String message) {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setMessage(message);
        else
            mProgressDialog = ProgressDialog.show(this, title, message, true, false);
    }

    private void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.dismiss();
    }

    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    public void showHorizontalProgressDialog(String title, String body) {

        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
        } else {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
            mProgressDialog.setIndeterminate(false);
            mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            mProgressDialog.setProgress(0);
            mProgressDialog.setMax(100);
            mProgressDialog.setCancelable(false);
            mProgressDialog.show();
        }
    }
}

```

```

public void updateProgress(int progress) {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.setProgress(progress);
    }
}

/**
* 第一步：创建存储
*
* @param view
*/
public void onCreateReferenceClick(View view) {
    mStorageReference = mFirebaseStorage.getReferenceFromUrl("gs://**something**.appspot.com");
    showToast("引用创建成功。");
    findViewById(R.id.button_step_2).setEnabled(true);
}

/**
* 第二步：创建名为“Images”的目录
*
* @param view
*/
public void onCreateDirectoryClick(View view) {
    mStorageReferenceImages = mStorageReference.child("images");
    showToast("目录 'images' 创建成功。");
    findViewById(R.id.button_step_3).setEnabled(true);
}

/**
* 第3步：上传图像文件并在ImageView上显示
*
* @param view
*/
public void onUploadFileClick(View view) {
    if (ContextCompat.checkSelfPermission(MainActivity.this,
        Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED ||
        ActivityCompat.checkSelfPermission(MainActivity.this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED)
        ActivityCompat.requestPermissions(MainActivity.this, new
            String[]{Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.WRITE_EXTERNAL_STORAGE},
            PERMISSION_READ_WRITE_EXTERNAL_STORAGE);
    else {
        pickImage();
    }
}

/**
* 第4步：下载图片文件并显示在ImageView上
*
* @param view
*/
public void onDownloadFileClick(View view) {
    downloadFile(mUri);
}

/**
* 第5步：删除图片文件并从ImageView中移除图片
*
* @param view
*/
public void onDeleteFileClick(View view) {
    deleteFile(mUri);
}

```

```

public void updateProgress(int progress) {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.setProgress(progress);
    }
}

/**
* Step 1: Create a Storage
*
* @param view
*/
public void onCreateReferenceClick(View view) {
    mStorageReference = mFirebaseStorage.getReferenceFromUrl("gs://**something**.appspot.com");
    showToast("Reference Created Successfully.");
    findViewById(R.id.button_step_2).setEnabled(true);
}

/**
* Step 2: Create a directory named "Images"
*
* @param view
*/
public void onCreateDirectoryClick(View view) {
    mStorageReferenceImages = mStorageReference.child("images");
    showToast("Directory 'images' created Successfully.");
    findViewById(R.id.button_step_3).setEnabled(true);
}

/**
* Step 3: Upload an Image File and display it on ImageView
*
* @param view
*/
public void onUploadFileClick(View view) {
    if (ContextCompat.checkSelfPermission(MainActivity.this,
        Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED ||
        ActivityCompat.checkSelfPermission(MainActivity.this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED)
        ActivityCompat.requestPermissions(MainActivity.this, new
            String[]{Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.WRITE_EXTERNAL_STORAGE},
            PERMISSION_READ_WRITE_EXTERNAL_STORAGE);
    else {
        pickImage();
    }
}

/**
* Step 4: Download an Image File and display it on ImageView
*
* @param view
*/
public void onDownloadFileClick(View view) {
    downloadFile(mUri);
}

/**
* Step 5: Delete an Image File and remove Image from ImageView
*
* @param view
*/
public void onDeleteFileClick(View view) {
    deleteFile(mUri);
}

```

```

}

private void showAlertDialog(Context ctx, String title, String body,
DialogInterface.OnClickListener okListener) {

    if (okListener == null) {
        okListener = new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        };
    }

    AlertDialog.Builder builder = new
        AlertDialog.Builder(ctx).setMessage(body).setPositiveButton("OK", okListener).setCancelable(false);

    if (!TextUtils.isEmpty(title)) {
        builder.setTitle(title);
    }

    builder.show();
}

private void uploadFile(Uri uri) {
    mImageView.setImageResource(R.drawable.placeholder_image);

    StorageReference uploadStorageReference =
mStorageReferenceImages.child(uri.getLastPathSegment());
    final UploadTask uploadTask = uploadStorageReference.putFile(uri);
    showHorizontalProgressDialog("Uploading", "Please wait...");
    uploadTask
.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        hideProgressDialog();
        Uri downloadUrl = taskSnapshot.getDownloadUrl();
        Log.d("MainActivity", downloadUrl.toString());
        showAlertDialog(MainActivity.this, "上传完成",
downloadUrl.toString(), new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                findViewById(R.id.button_step_3).setEnabled(false);
                findViewById(R.id.button_step_4).setEnabled(true);
            }
        });
    }

    Glide.with(MainActivity.this)
        .load(downloadUrl)
        .into(mImageView);
}

.addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        exception.printStackTrace();
        // 处理上传失败
        hideProgressDialog();
    }
})
.addOnProgressListener(MainActivity.this, new
OnProgressListener<UploadTask.TaskSnapshot>() {
}
}

private void showAlertDialog(Context ctx, String title, String body,
DialogInterface.OnClickListener okListener) {

    if (okListener == null) {
        okListener = new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        };
    }

    AlertDialog.Builder builder = new
        AlertDialog.Builder(ctx).setMessage(body).setPositiveButton("OK", okListener).setCancelable(false);

    if (!TextUtils.isEmpty(title)) {
        builder.setTitle(title);
    }

    builder.show();
}

private void uploadFile(Uri uri) {
    mImageView.setImageResource(R.drawable.placeholder_image);

    StorageReference uploadStorageReference =
mStorageReferenceImages.child(uri.getLastPathSegment());
    final UploadTask uploadTask = uploadStorageReference.putFile(uri);
    showHorizontalProgressDialog("Uploading", "Please wait...");
    uploadTask
        .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                hideProgressDialog();
                Uri downloadUrl = taskSnapshot.getDownloadUrl();
                Log.d("MainActivity", downloadUrl.toString());
                showAlertDialog(MainActivity.this, "Upload Complete",
downloadUrl.toString(), new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        findViewById(R.id.button_step_3).setEnabled(false);
                        findViewById(R.id.button_step_4).setEnabled(true);
                    }
                });

                Glide.with(MainActivity.this)
                    .load(downloadUrl)
                    .into(mImageView);
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                exception.printStackTrace();
                // Handle unsuccessful uploads
                hideProgressDialog();
            }
        })
        .addOnProgressListener(MainActivity.this, new
OnProgressListener<UploadTask.TaskSnapshot>() {
}
}
}

```

```

@Override
public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
    int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
Log.i("Progress", progress + "");
    updateProgress(progress);
}
});

private void downloadFile(Uri uri) {
    mImageView.setImageResource(R.drawable.placeholder_image);
    final StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), "Firebase Storage");
    if (!mediaStorageDir.exists()) {
        if (!mediaStorageDir.mkdirs()) {
Log.d("MainActivity", "failed to create Firebase Storage directory");
    }
}

final File localFile = new File(mediaStorageDir, uri.getLastPathSegment());
try {
localFile.createNewFile();
} catch (IOException e) {
e.printStackTrace();
}

showHorizontalProgressDialog("下载中", "请稍候...");
storageReferenceImage.getFile(localFile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
        hideProgressDialog();
showAlertDialog(MainActivity.this, "下载完成",
localFile.getAbsolutePath(), new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        findViewById(R.id.button_step_4).setEnabled(false);
        findViewById(R.id.button_step_5).setEnabled(true);
    }
});

Glide.with(MainActivity.this)
    .load(localFile)
    .into(mImageView);
}
).addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception exception) {
    // 处理任何错误
hideProgressDialog();
    exception.printStackTrace();
}
}).addOnProgressListener(new OnProgressListener<FileDownloadTask.TaskSnapshot>() {
@Override
public void onProgress(FileDownloadTask.TaskSnapshot taskSnapshot) {
    int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
Log.i("进度", progress + "");
    updateProgress(progress);
}
});
}

```

```

@Override
public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
    int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
Log.i("Progress", progress + "");
    updateProgress(progress);
}
});

private void downloadFile(Uri uri) {
    mImageView.setImageResource(R.drawable.placeholder_image);
    final StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), "Firebase Storage");
    if (!mediaStorageDir.exists()) {
        if (!mediaStorageDir.mkdirs()) {
Log.d("MainActivity", "failed to create Firebase Storage directory");
    }
}

final File localFile = new File(mediaStorageDir, uri.getLastPathSegment());
try {
    localFile.createNewFile();
} catch (IOException e) {
    e.printStackTrace();
}

showHorizontalProgressDialog("Downloading", "Please wait...");
storageReferenceImage.getFile(localFile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
        hideProgressDialog();
        showAlertDialog(MainActivity.this, "Download Complete",
localFile.getAbsolutePath(), new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                findViewById(R.id.button_step_4).setEnabled(false);
                findViewById(R.id.button_step_5).setEnabled(true);
            }
        });

Glide.with(MainActivity.this)
    .load(localFile)
    .into(mImageView);
}
).addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception exception) {
    // Handle any errors
hideProgressDialog();
    exception.printStackTrace();
}
}).addOnProgressListener(new OnProgressListener<FileDownloadTask.TaskSnapshot>() {
@Override
public void onProgress(FileDownloadTask.TaskSnapshot taskSnapshot) {
    int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
Log.i("Progress", progress + "");
    updateProgress(progress);
}
});
}

```

```

    });

}

private void deleteFile(Uri uri) {
    showProgressDialog("删除中", "请稍候...");
    StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
    storageReferenceImage.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            hideProgressDialog();
            showAlertDialog(MainActivity.this, "成功", "文件删除成功。", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    mImageView.setImageResource(R.drawable.placeholder_image);
                    findViewById(R.id.button_step_3).setEnabled(true);
                    findViewById(R.id.button_step_4).setEnabled(false);
                    findViewById(R.id.button_step_5).setEnabled(false);
                }
            });
            File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_PICTURES), "Firebase Storage");
            if (!mediaStorageDir.exists()) {
                if (!mediaStorageDir.mkdirs()) {
Log.d("MainActivity", "创建 Firebase 存储目录失败");
                }
            }
            deleteFiles(mediaStorageDir);
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            hideProgressDialog();
            exception.printStackTrace();
        }
    });
}

private void deleteFiles(File directory) {
    if (directory.isDirectory())
        for (File child : directory.listFiles())
            child.delete();
}
}

```

默认情况下，Firebase Storage 规则应用身份验证限制。只有用户通过身份验证后，才能对 Firebase Storage 执行操作，否则不能。我在此演示中通过更新存储规则禁用了身份验证部分。之前，规则如下所示：

```

service firebase.storage {
    match /b/**something**.appspot.com/o {
        match /{allPaths=**} {
allow read, write: if request.auth != null;
    }
}
}

```

但我改为跳过身份验证：

```

    });

}

private void deleteFile(Uri uri) {
    showProgressDialog("Deleting", "Please wait...");
    StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
    storageReferenceImage.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            hideProgressDialog();
            showAlertDialog(MainActivity.this, "Success", "File deleted successfully.", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    mImageView.setImageResource(R.drawable.placeholder_image);
                    findViewById(R.id.button_step_3).setEnabled(true);
                    findViewById(R.id.button_step_4).setEnabled(false);
                    findViewById(R.id.button_step_5).setEnabled(false);
                }
            });
            File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_PICTURES), "Firebase Storage");
            if (!mediaStorageDir.exists()) {
                if (!mediaStorageDir.mkdirs()) {
Log.d("MainActivity", "failed to create Firebase Storage directory");
                }
            }
            deleteFiles(mediaStorageDir);
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            hideProgressDialog();
            exception.printStackTrace();
        }
    });
}

private void deleteFiles(File directory) {
    if (directory.isDirectory())
        for (File child : directory.listFiles())
            child.delete();
}
}

```

By default, Firebase Storage rules applies Authentication restriction. If user is authenticated, only then, he can perform operations on Firebase Storage, else he cannot. I have disabled the authentication part in this demo by updating Storage rules. Previously, rules were looking like:

```

service firebase.storage {
    match /b/**something**.appspot.com/o {
        match /{allPaths=**} {
allow read, write: if request.auth != null;
        }
    }
}

```

But I changed to skip the authentication:

```

service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      // 允许读取, 写入;
    }
}

```

第229.7节：Firebase实时数据库：如何设置/获取数据

注意：让我们为示例设置一些匿名身份验证

```

{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}

```

完成后，通过编辑数据库地址创建一个子节点。例如：

<https://your-project.firebaseio.com/> 到 <https://your-project.firebaseio.com/chat>

我们将从Android设备向此位置写入数据。你不必创建数据库结构（标签、字段等），当你发送Java对象到Firebase时，它会自动创建！

创建一个包含所有要发送到数据库的属性的Java对象：

```

public class ChatMessage {
  private String 用户名;
  private String 消息;

  public ChatMessage(String 用户名, String 消息) {
    this.用户名 = 用户名;
    this.消息 = 消息;
  }

  public ChatMessage() {} // 你必须有一个空构造函数

  public String 获得用户名() {
    return 用户名;
  }

  public String 获得消息() {
    return 消息;
  }
}

```

然后在你的活动中：

```

if (FirebaseAuth.getInstance().getCurrentUser() == null) {
  FirebaseAuth.getInstance().匿名登录().addOnCompleteListener(new
  OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> 任务) {
      if (任务.isComplete() && 任务.isSuccessful()){
        FirebaseDatabase 数据库 = FirebaseDatabase.getInstance();
      }
    }
}

```

```

service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}

```

Section 229.7: Firebase Realtime Database: how to set/get data

Note: Let's setup some anonymous authentication for the example

```

{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}

```

Once it is done, create a child by editing your database address. For example:

<https://your-project.firebaseio.com/> to <https://your-project.firebaseio.com/chat>

We will put data to this location from our Android device. You **don't have to** create the database structure (tabs, fields... etc), it will be automatically created when you'll send Java object to Firebase!

Create a Java object that contains all the attributes you want to send to the database:

```

public class ChatMessage {
  private String username;
  private String message;

  public ChatMessage(String username, String message) {
    this.username = username;
    this.message = message;
  }

  public ChatMessage() {} // you MUST have an empty constructor

  public String getUsername() {
    return username;
  }

  public String getMessage() {
    return message;
  }
}

```

Then in your activity:

```

if (FirebaseAuth.getInstance().getCurrentUser() == null) {
  FirebaseAuth.getInstance().signInAnonymously().addOnCompleteListener(new
  OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
      if (task.isComplete() && task.isSuccessful()){
        FirebaseDatabase database = FirebaseDatabase.getInstance();
      }
    }
}

```

```
DatabaseReference reference = database.getReference("chat"); // reference 是  
'chat', 因为我们在 /chat 创建了数据库  
        }  
    }  
});  
}
```

发送一个值

```
ChatMessage msg = new ChatMessage("user1", "Hello World!")  
reference.push().setValue(msg);
```

接收数据库中发生的变化：

```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```



第229.8节：基于FCM通知的演示

此示例展示了如何使用 Firebase 云消息传递 (FCM) 平台。FCM 是 Google 云消息传递 (GCM) 的继任者。它不需要应用用户的 C2D_MESSAGE 权限。

整合FCM的步骤如下。

1. 在 Android Studio 中创建示例 Hello World 项目，您的 Android Studio 界面将如下图所示

```
        DatabaseReference reference = database.getReference("chat"); // reference is  
'chat' because we created the database at /chat  
    }  
}  
});  
}  
}
```

To send a value:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

To receive changes that occurs in the database:

```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

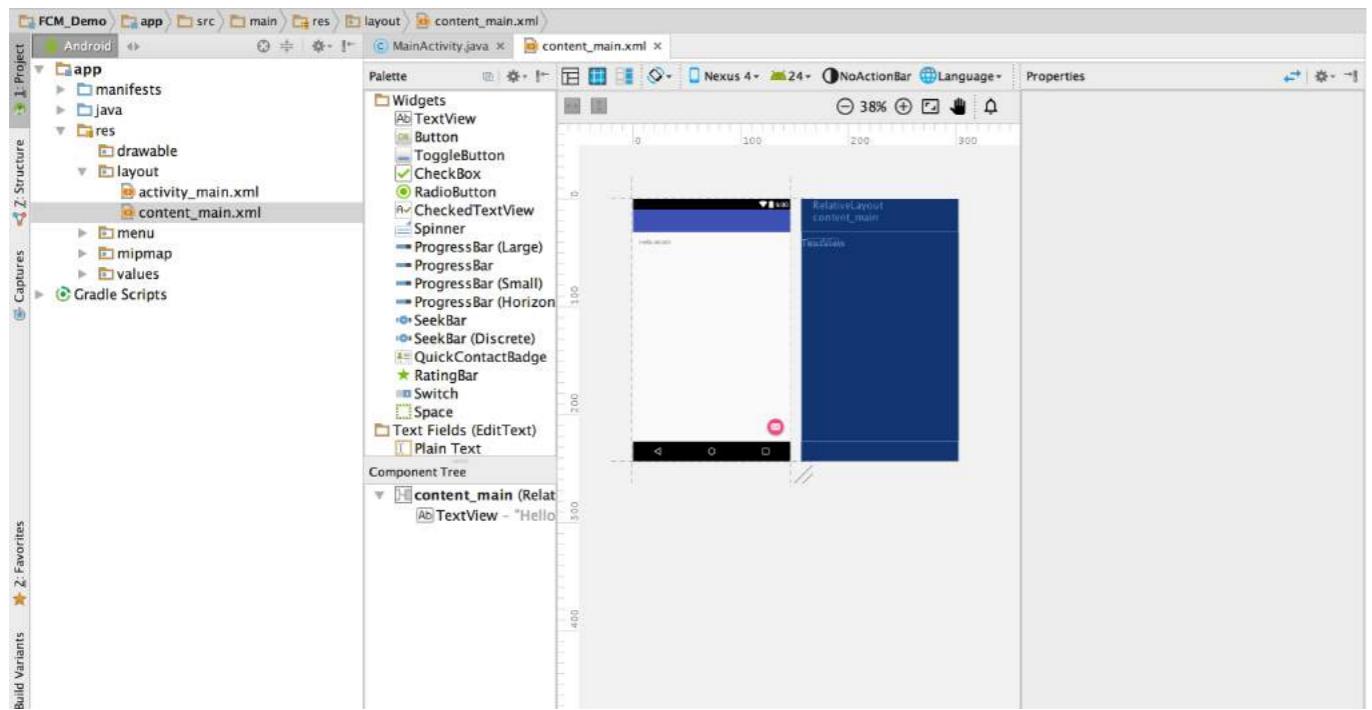


Section 229.8: Demo of FCM based notifications

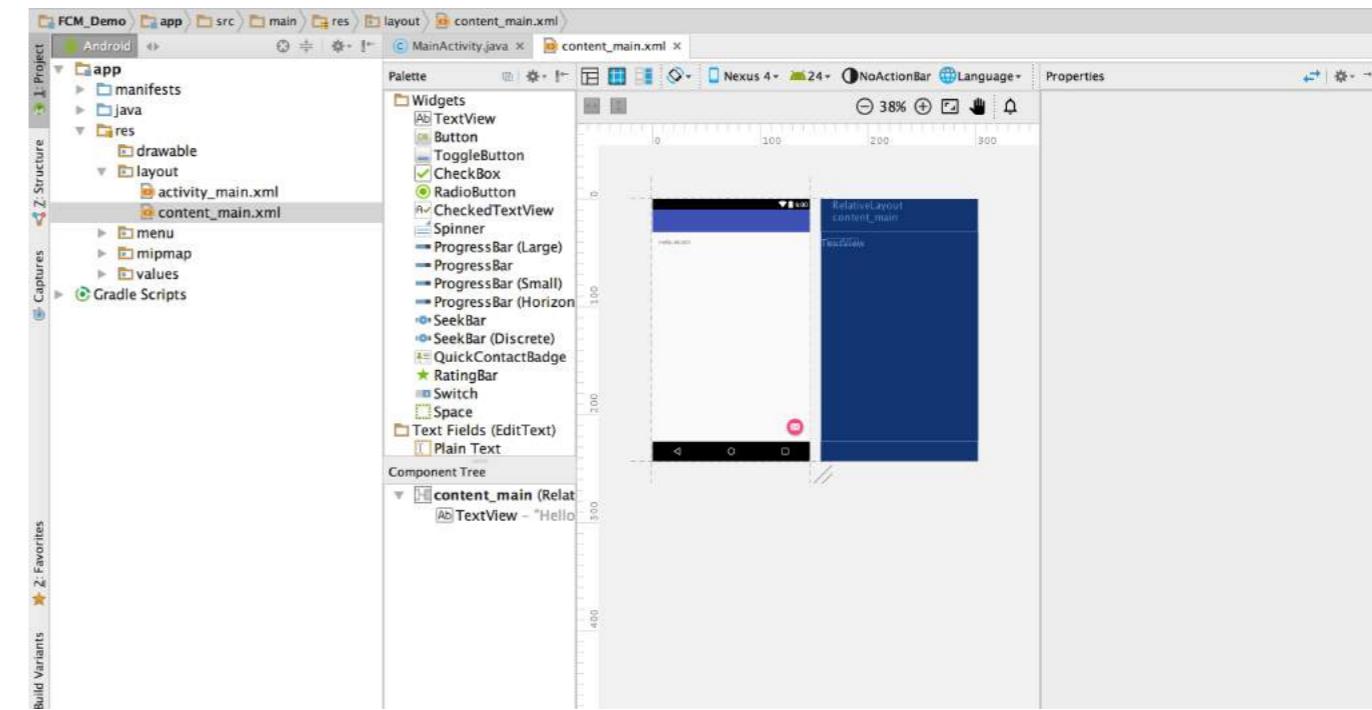
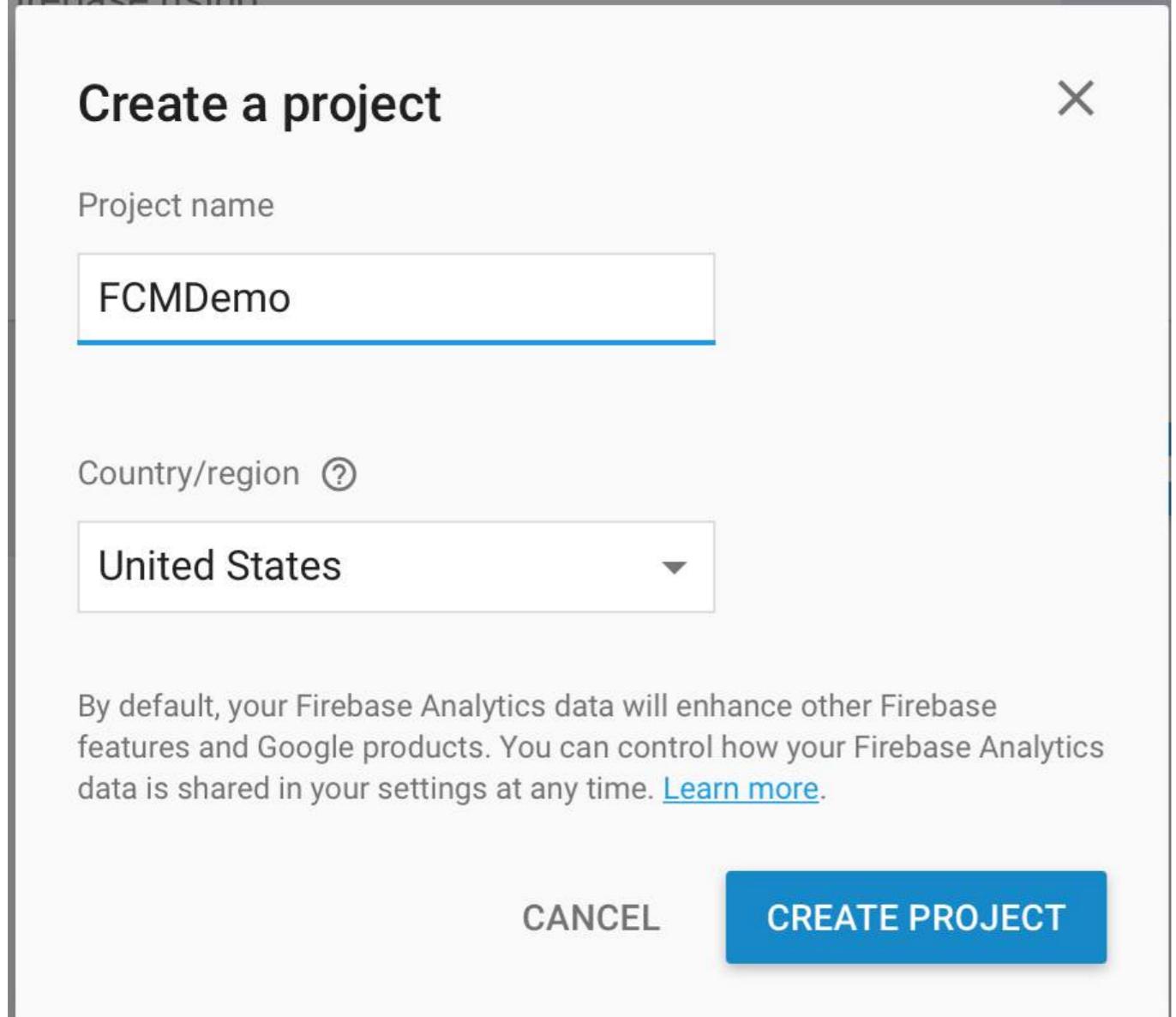
This example shows how to use the Firebase Cloud Messaging(FCM) platform. FCM is a successor of Google Cloud Messaging(GCM). It does not require C2D_MESSAGE permissions from the app users.

Steps to integrate FCM are as follows.

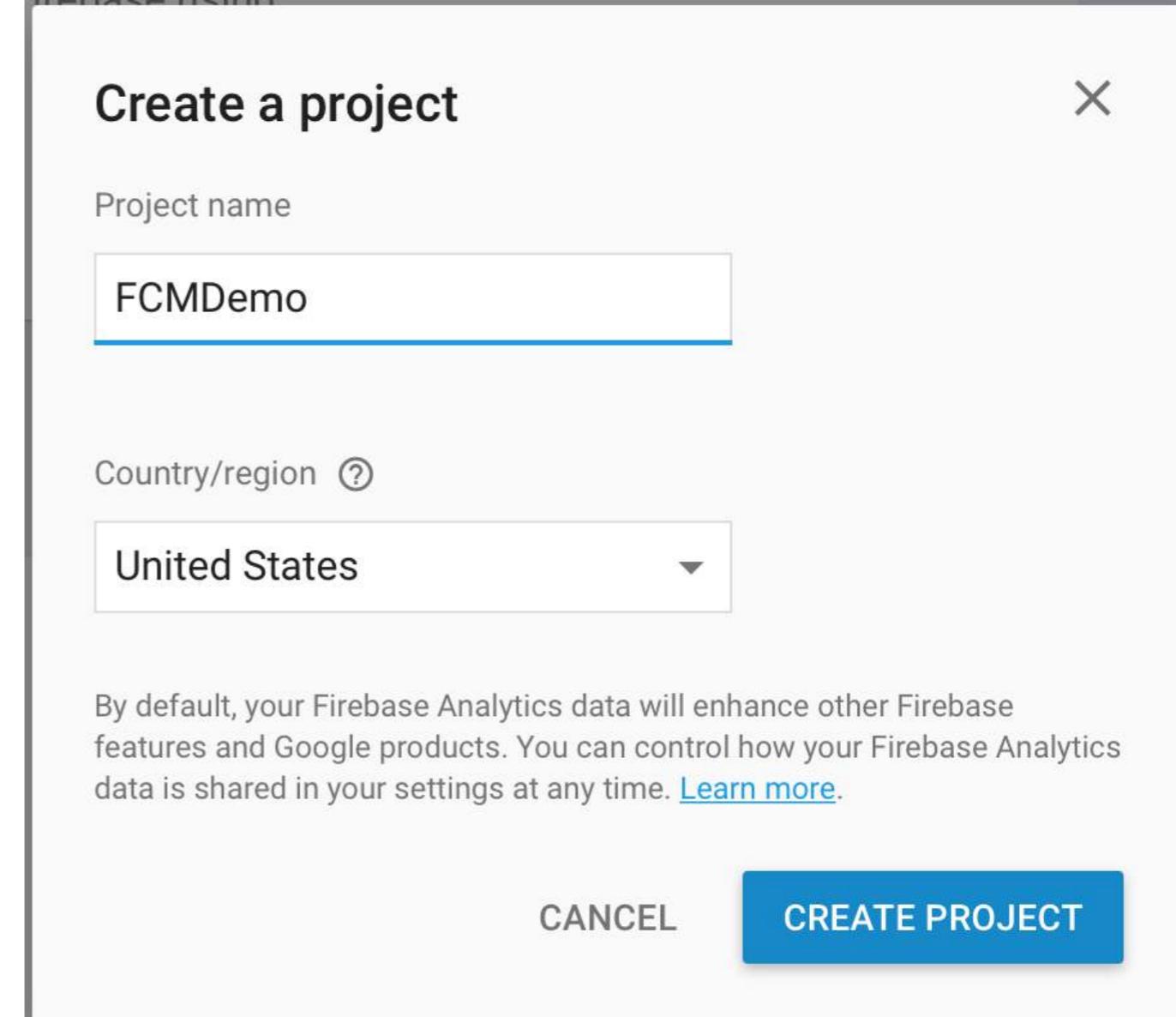
1. Create sample hello world project in Android Studio Your Android studio screen would look like the following picture.



2. 下一步是设置 Firebase 项目。访问 <https://console.firebaseio.google.com> 并创建一个项目，相同的名称，便于您轻松跟踪。



2. Next step is to set up firebase project. Visit <https://console.firebaseio.google.com> and create a project with an identical name, so that you can track it easily.



3. 现在是时候将 Firebase 添加到您刚创建的示例安卓项目中了。您将需要项目的包名和调试签名证书的 SHA-1 (可选)。

- a. 包名 - 可以从 Android 清单 XML 文件中找到。
- b. 调试签名 SHA-1 证书 - 可以通过在终端运行以下命令找到。

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

在 Firebase 控制台输入此信息并将应用添加到 Firebase 项目中。点击添加应用按钮后，浏览器会自动下载一个名为 "google-services.json" 的 JSON 文件。

4. 现在将刚下载的 google-services.json 文件复制到你的 Android 应用模块根目录中。

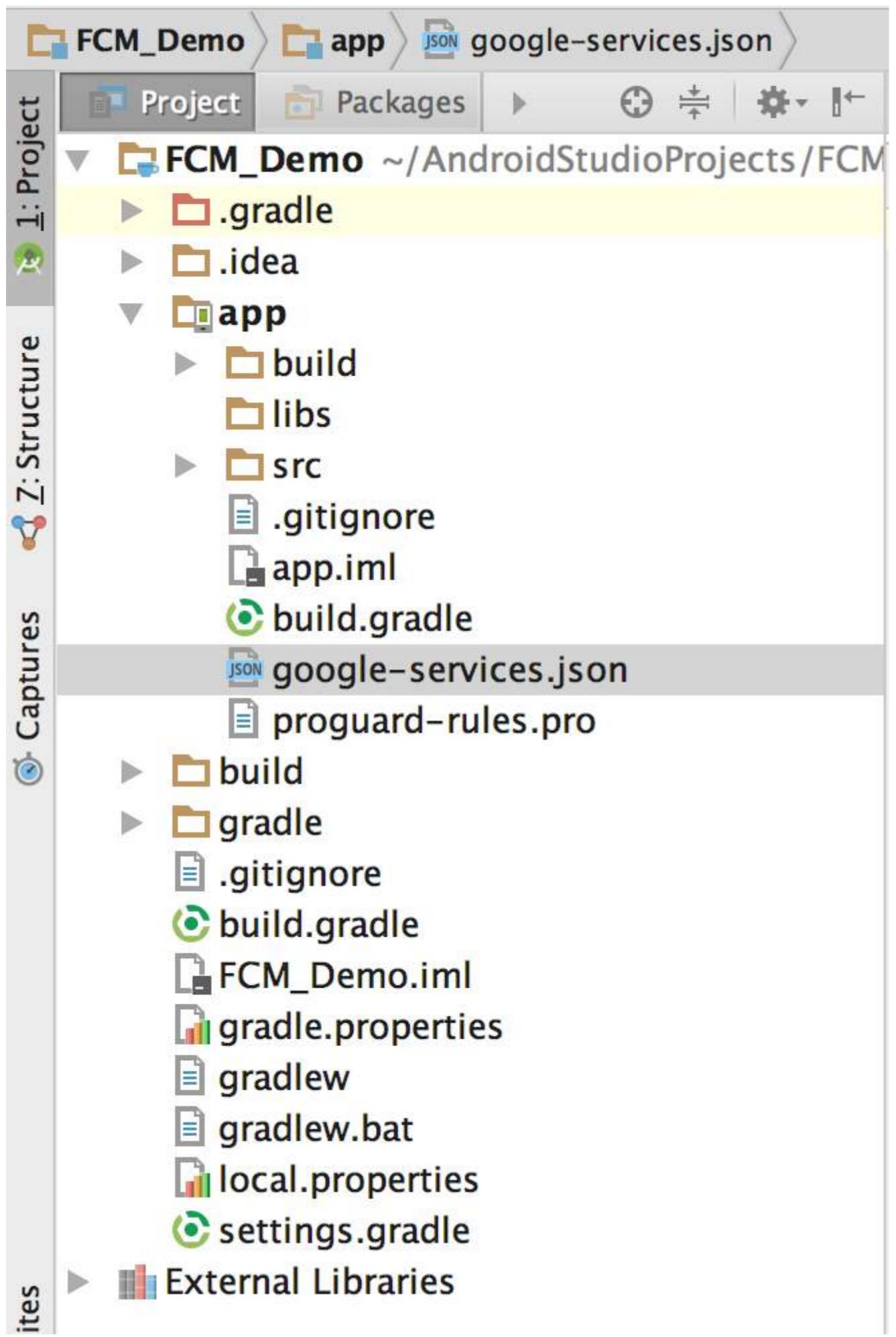
3. Now it is time to add firebase to your sample android project you have just created. You will need package name of your project and Debug signing certificate SHA-1(optional).

- a. Package name - It can be found from the android manifest XML file.
- b. Debug signing SHA-1 certificate - It can be found by running following command in the terminal.

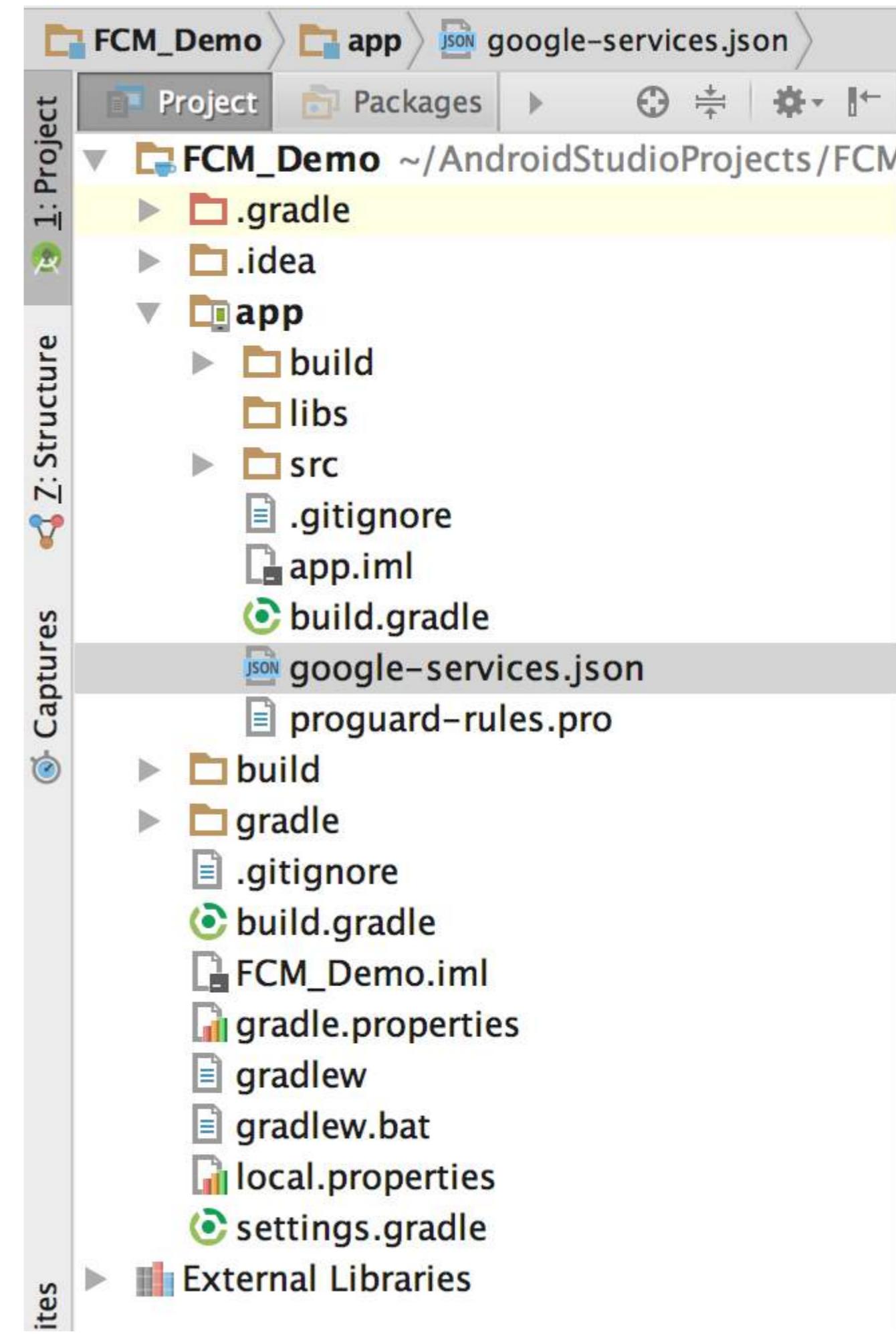
```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

Enter this information in the firebase console and add the app to firebase project. Once you click on add app button, your browser would automatically download a JSON file named "google-services.json".

4. Now copy the google-services.json file you have just downloaded into your Android app module root directory.



5. 按照 Firebase 控制台上的指示继续操作。a. 在你的项目级 build.gradle 中添加以下代码行



5. Follow the instructions given on the firebase console as you proceed ahead. a. Add following code line to

你的项目级 build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' ....}
```

b. 在你的应用级 build.gradle 文件末尾添加以下代码行。

```
// 以下是需要添加的依赖项  
compile 'com.google.firebaseio:messaging:11.0.4'  
compile 'com.android.support:multidex:1.0.1'  
}  
// 该行直到文件末尾  
apply plugin: 'com.google.gms.google-services'
```

c. Android Studio 会提示你同步项目。点击“立即同步”。

6. 下一步任务是添加两个服务。a. 一个继承自 FirebaseMessagingService，带有如下 intent-filter

```
<intent-filter>  
    <action android:name="com.google.firebaseio.MESSAGING_EVENT"/>  
</intent-filter>
```

b. 一个继承自 FirebaseInstanceIdService。

```
<intent-filter>  
    <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT"/>  
</intent-filter>
```

7. FirebaseMessagingService 的代码应如下所示。

```
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
  
import com.google.firebaseio.messaging.FirebaseMessagingService;  
  
public class MyFirebaseMessagingService extends FirebaseMessagingService {  
    public MyFirebaseMessagingService() {  
    }  
}
```

8. FirebaseInstanceIdService 应如下所示。

```
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
  
import com.google.firebaseio.iid.FirebaseInstanceIdService;  
  
public class MyFirebaseInstanceIdService extends FirebaseInstanceIdService {  
    public MyFirebaseInstanceIdService() {  
    }  
}
```

9. 现在是获取设备注册令牌的时候了。将以下代码行添加到 MainActivity 的 onCreate 方法中。

your project level build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' ....}
```

b. Add following code line at the end of your app level build.gradle.

```
//following are the dependencies to be added  
compile 'com.google.firebaseio:messaging:11.0.4'  
compile 'com.android.support:multidex:1.0.1'  
}  
// this line goes to the end of the file  
apply plugin: 'com.google.gms.google-services'
```

c. Android studio would ask you to sync project. Click on Sync now.

6. Next task is to add two services. a. One extending FirebaseMessagingService with intent-filter as following

```
<intent-filter>  
    <action android:name="com.google.firebaseio.MESSAGING_EVENT"/>  
</intent-filter>
```

b. One extending FirebaseInstanceIdService.

```
<intent-filter>  
    <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT"/>  
</intent-filter>
```

7. FirebaseMessagingService code should look like this.

```
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
  
import com.google.firebaseio.messaging.FirebaseMessagingService;  
  
public class MyFirebaseMessagingService extends FirebaseMessagingService {  
    public MyFirebaseMessagingService() {  
    }  
}
```

8. FirebaseInstanceIdService should look like this.

```
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
  
import com.google.firebaseio.iid.FirebaseInstanceIdService;  
  
public class MyFirebaseInstanceIdService extends FirebaseInstanceIdService {  
    public MyFirebaseInstanceIdService() {  
    }  
}
```

9. Now it is time to capture the device registration token. Add following line of code to MainActivity's onCreate method.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. 一旦我们获得了访问令牌，就可以使用 Firebase 控制台发送通知。在

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Once we have the access token, we can use firebase console to send out the notification. Run the app on

你的安卓手机上运行该应用。

your android handset.



Firebase



FCMDemo



Analytics

DEVELOP



Auth



Database



Storage



Hosting



Remote Config



Test Lab



Crash

GROW



Notifications



Dynamic Links

FADNI



Firebase



FCMDemo



Analytics

DEVELOP



Auth



Database



Storage



Hosting



Remote Config



Test Lab



Crash

GROW



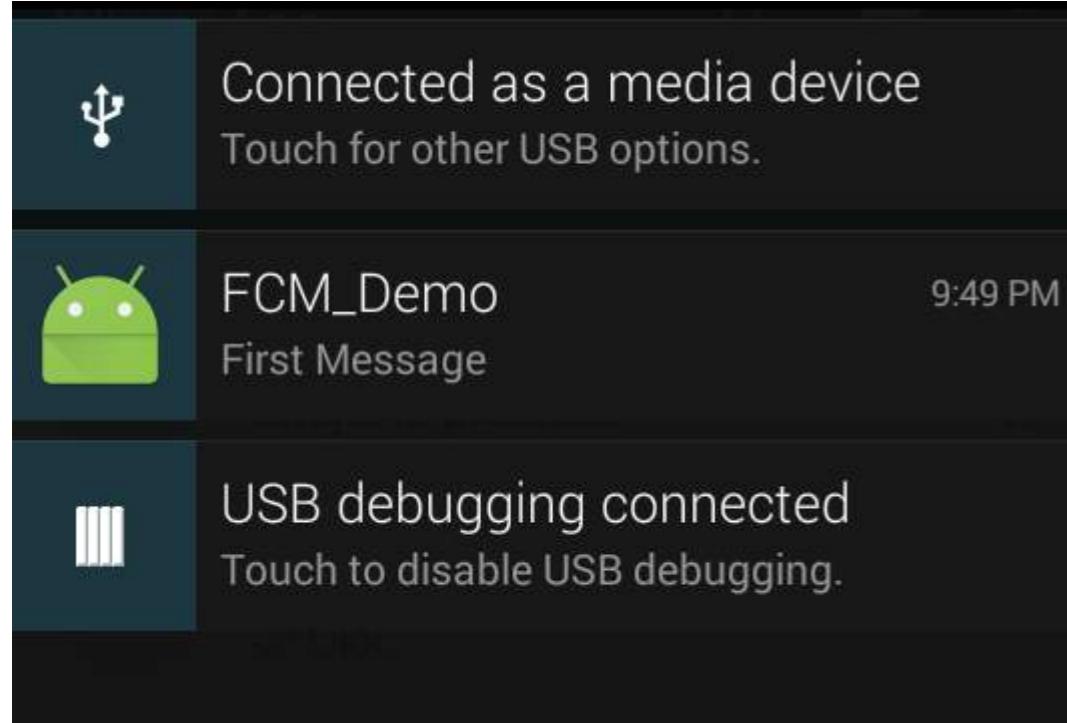
Notifications



Dynamic Links

FADNI

点击 Firebase 控制台中的通知，界面将帮助您发送第一条消息。Firebase 提供了向单个设备发送消息的功能（通过我们捕获的设备令牌 ID），也可以向使用我们应用的所有用户或特定用户组发送消息。发送第一条消息后，您的手机屏幕应如下所示。



谢谢

第 229.9 节：使用邮箱和密码登录 Firebase 用户

```
public class LoginActivity extends AppCompatActivity {

    @BindView(R.id.tIETLoginEmail)
    EditText mEditTextEmail;
    @BindView(R.id.tIETLoginPassword)
    EditText mEditTextPassword;

    @Override
    protected void onResume() {
        super.onResume();
        FirebaseUser firebaseUser = mAuth.getCurrentUser();
        if (firebaseUser != null)
            startActivity(new Intent(this, HomeActivity.class));
    }

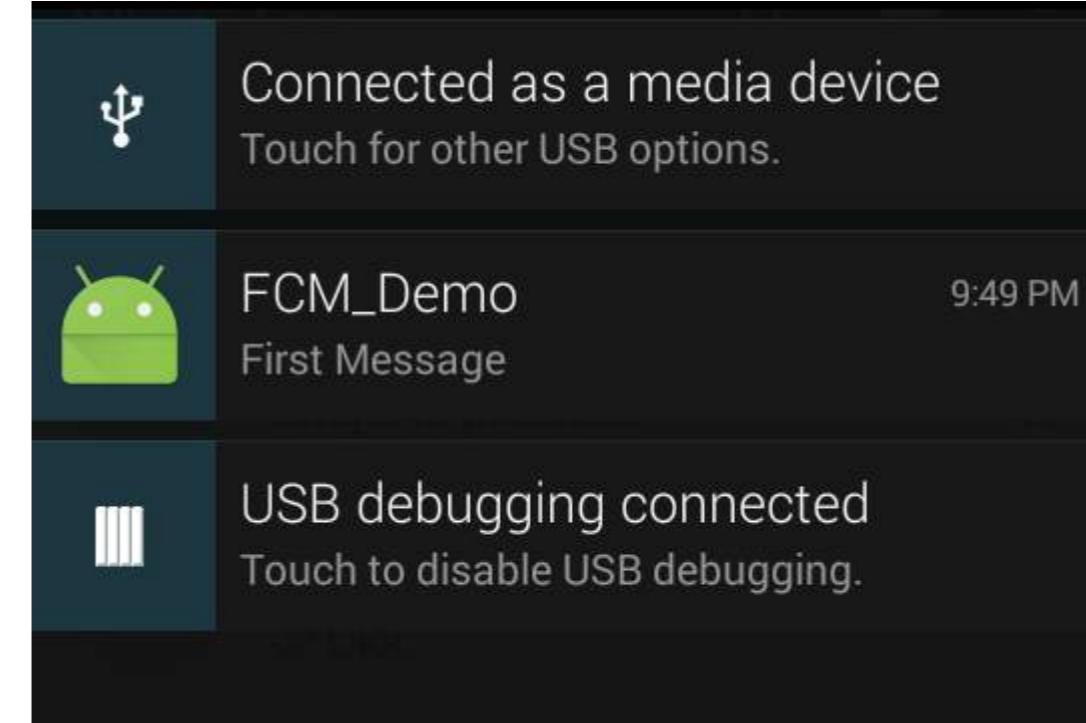
    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_login;
    }

    @OnClick(R.id.btnLoginLogin)
    void onSignInClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "请输入邮箱");
            return;
        }
    }
}
```

Click on Notification in Firebase console and UI will help you to send out your first message. Firebase offers functionality to send messages to single device(By using the device token id we captured) or all the users using our app or to specific group of users. Once you send your first message, your mobile screen should look like following.



Thank you

Section 229.9: Sign In Firebase user with email and password

```
public class LoginActivity extends AppCompatActivity {

    @BindView(R.id.tIETLoginEmail)
    EditText mEditTextEmail;
    @BindView(R.id.tIETLoginPassword)
    EditText mEditTextPassword;

    @Override
    protected void onResume() {
        super.onResume();
        FirebaseUser firebaseUser = mAuth.getCurrentUser();
        if (firebaseUser != null)
            startActivity(new Intent(this, HomeActivity.class));
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_login;
    }

    @OnClick(R.id.btnLoginLogin)
    void onSignInClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
        }
    }
}
```

```

if (!FormValidationUtils.isEmailValid(mEditEmail)) {
    FormValidationUtils.setError(null, mEditEmail, "请输入有效的邮箱");
    return;
}

if (TextUtils.isEmpty(mEditPassword.getText())) {
    FormValidationUtils.setError(null, mEditPassword, "请输入密码");
    return;
}

signInWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
}

private void signInWithEmailAndPassword(String email, String password) {
    DialogUtils.showProgressDialog(this, "", getString(R.string.sign_in), false);
    mFirebaseAuth
.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            DialogUtils.dismissProgressDialog();

            if (task.isSuccessful()) {
                Toast.makeText(LoginActivity.this, "登录成功",
Toast.LENGTH_SHORT).show();
                startActivity(new Intent(LoginActivity.this, HomeActivity.class));
                finish();
            } else {
                Toast.makeText(LoginActivity.this, task.getException().getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
}
}

@OnClick(R.id.btnLoginSignUp)
void onSignUpClick() {
startActivity(new Intent(this, SignUpActivity.class));
}
}

@OnClick(R.id.btnLoginForgotPassword)
void forgotPassword() {
startActivity(new Intent(this, ForgotPasswordActivity.class));
}
}

```

第229.10节：发送Firebase密码重置邮件

```

public class ForgotPasswordActivity extends AppCompatActivity {

    @BindView(R.id.tIETForgotPasswordEmail)
EditText mEditEmail;
private FirebaseAuth mFirebaseAuth;
private FirebaseAuth.AuthStateListener mAuthStateListener;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
}

```

```

if (!FormValidationUtils.isEmailValid(mEditEmail)) {
    FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
    return;
}

if (TextUtils.isEmpty(mEditPassword.getText())) {
    FormValidationUtils.setError(null, mEditPassword, "Please enter password");
    return;
}

signInWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
}

private void signInWithEmailAndPassword(String email, String password) {
    DialogUtils.showProgressDialog(this, "", getString(R.string.sign_in), false);
    mFirebaseAuth
.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            DialogUtils.dismissProgressDialog();

            if (task.isSuccessful()) {
                Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();
                startActivity(new Intent(LoginActivity.this, HomeActivity.class));
                finish();
            } else {
                Toast.makeText(LoginActivity.this, task.getException().getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
}
}

@OnClick(R.id.btnLoginForgotPassword)
void forgotPassword() {
    startActivity(new Intent(this, ForgotPasswordActivity.class));
}
}

```

Section 229.10: Send Firebase password reset email

```

public class ForgotPasswordActivity extends AppCompatActivity {

    @BindView(R.id.tIETForgotPasswordEmail)
EditText mEditEmail;
private FirebaseAuth mFirebaseAuth;
private FirebaseAuth.AuthStateListener mAuthStateListener;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
}

```

```

setContentView(R.layout.activity_forgot_password);
ButterKnife.bind(this);

mFirebaseAuth = FirebaseAuth.getInstance();

mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
        if (firebaseUser != null) {
            // 你可以通过 firebaseUser.getUid() 对 UserId 进行任何操作
        } else {
        }
    }
};

@Override
protected void onStart() {
    super.onStart();
mFirebaseAuth.addAuthStateListener(mAuthStateListener);
}

@Override
protected void onStop() {
    super.onStop();
    if (mAuthStateListener != null) {
mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
    }
}

@OnClick(R.id.btnForgotPasswordSubmit)
void onSubmitClick() {

    if (FormValidationUtils.isBlank(mEditText)) {
        FormValidationUtils.setError(null, mEditText, "请输入邮箱");
        return;
    }

    if (!FormValidationUtils.isEmailValid(mEditText)) {
        FormValidationUtils.setError(null, mEditText, "请输入有效的邮箱");
        return;
    }

DialogUtils.showProgressDialog(this, "", "请稍候...", false);
mFirebaseAuth.sendPasswordResetEmail(mEditText.getText().toString())
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
Toast.makeText(ForgotPasswordActivity.this, "一封邮件已发送到您的邮箱。", Toast.LENGTH_SHORT).show();
finish();
        } else {
Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
}

```

```

setContentView(R.layout.activity_forgot_password);
ButterKnife.bind(this);

mFirebaseAuth = FirebaseAuth.getInstance();

mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
        if (firebaseUser != null) {
            // Do whatever you want with the UserId by firebaseUser.getUid()
        } else {
        }
    }
};

@Override
protected void onStart() {
    super.onStart();
    mFirebaseAuth.addAuthStateListener(mAuthStateListener);
}

@Override
protected void onStop() {
    super.onStop();
    if (mAuthStateListener != null) {
        mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
    }
}

@OnClick(R.id.btnForgotPasswordSubmit)
void onSubmitClick() {

    if (FormValidationUtils.isBlank(mEditText)) {
        FormValidationUtils.setError(null, mEditText, "Please enter email");
        return;
    }

    if (!FormValidationUtils.isEmailValid(mEditText)) {
        FormValidationUtils.setError(null, mEditText, "Please enter valid email");
        return;
    }

DialogUtils.showProgressDialog(this, "", "Please wait...", false);
mFirebaseAuth.sendPasswordResetEmail(mEditText.getText().toString())
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            DialogUtils.dismissProgressDialog();
            if (task.isSuccessful()) {
                Toast.makeText(ForgotPasswordActivity.this, "An email has been sent to
you.", Toast.LENGTH_SHORT).show();
                finish();
            } else {
                Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    });
}
}

```

}

第229.11节：重新验证Firebase用户

```

public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "请输入邮箱");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "请输入有效的邮箱");
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "请输入密码");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
                           mEditTextPassword.getText().toString());
    }

    private void 重新认证用户(String 邮箱, String 密码) {
        DialogUtils.显示进度对话框(获取活动(), "重新认证中", "请稍候...", false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    DialogUtils.dismissProgressDialog();
                    if (task.isSuccessful()) {
                        mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                        dismiss();
                    } else {
                        ((BaseAppCompatActivity)
                            getActivity()).showToast(task.getException().getMessage());
                    }
                }
            });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
    }
}

```

}

Section 229.11: Re-Authenticate Firebase user

```

public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "Please enter password");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
                           mEditTextPassword.getText().toString());
    }

    private void reauthenticateUser(String email, String password) {
        DialogUtils.showProgressDialog(getActivity(), "Re-Authenticating", "Please wait...", false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    DialogUtils.dismissProgressDialog();
                    if (task.isSuccessful()) {
                        mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                        dismiss();
                    } else {
                        ((BaseAppCompatActivity)
                            getActivity()).showToast(task.getException().getMessage());
                    }
                }
            });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
    }
}

```

```

}

@OnClick(R.id.btn_dialog_reauthenticate_cancel)
void onCancelClick() {
    dismiss();
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
    ButterKnife.bind(this, view);
    return view;
}

@Override
public void onResume() {
    super.onResume();
    Window window = getDialog().getWindow();
    window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
                    WindowManager.LayoutParams.WRAP_CONTENT);
}

interface OnReauthenticateSuccessListener {
    void onReauthenticateSuccess();
}
}

```

第229.12节：Firebase 登出

变量初始化

```
private GoogleApiClient mGoogleApiClient;
```

当你放置登出按钮时，必须在所有的 onCreate() 方法中编写这段代码。

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
    .addApi(Auth.GOOGLE_SIGN_IN_API)
.build();
```

将以下代码放在登出按钮上。

```
Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
    new ResultCallback<Status>() {
        @Override
        public void onResult(Status status) {
            FirebaseAuth.getInstance().signOut();
            Intent i1 = new Intent(MainActivity.this, GoogleSignInActivity.class);
            startActivity(i1);
            Toast.makeText(MainActivity.this, "注销成功！", Toast.LENGTH_SHORT).show();
        }
    });

```

```

}

@OnClick(R.id.btn_dialog_reauthenticate_cancel)
void onCancelClick() {
    dismiss();
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
    ButterKnife.bind(this, view);
    return view;
}

@Override
public void onResume() {
    super.onResume();
    Window window = getDialog().getWindow();
    window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
                    WindowManager.LayoutParams.WRAP_CONTENT);
}

interface OnReauthenticateSuccessListener {
    void onReauthenticateSuccess();
}
}

```

Section 229.12: Firebase Sign Out

Initialization of variable

```
private GoogleApiClient mGoogleApiClient;
```

You must have to Write this Code in onCreate() method of all that when u put signout button.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
    .addApi(Auth.GOOGLE_SIGN_IN_API)
.build();
```

Put below code on signout button.

```
Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
    new ResultCallback<Status>() {
        @Override
        public void onResult(Status status) {
            FirebaseAuth.getInstance().signOut();
            Intent i1 = new Intent(MainActivity.this, GoogleSignInActivity.class);
            startActivity(i1);
            Toast.makeText(MainActivity.this, "Logout Successfully!", Toast.LENGTH_SHORT).show();
        }
    });

```

第230章：Firebase云消息传递

Firebase云消息传递（FCM）是一种跨平台消息解决方案，允许您可靠地免费发送消息。

使用 FCM，您可以通知客户端应用有新的电子邮件或其他数据可供同步。您可以发送通知消息以促进用户重新参与和留存。对于即时通讯等用例，消息可以传输最多 4KB 的有效载荷到客户端应用。

第 230.1 节：在 Android 上设置 Firebase 云消息传递客户端应用

- 完成安装和设置部分，将您的应用连接到 Firebase。

这将会在 Firebase 中创建项目。

- 将 Firebase 云消息传递的依赖项添加到您的模块级 build.gradle 文件中：

```
dependencies {  
    compile 'com.google.firebaseio:firebase-messaging:10.2.1'  
}
```

现在您已准备好在 Android 中使用 FCM。

FCM 客户端需要运行 Android 2.3 或更高版本且安装了 Google Play 商店应用的设备，或运行带有 Google API 的 Android 2.3 模拟器。

编辑您的 AndroidManifest.xml 文件

```
<service  
    android:name=".MyFirebaseMessagingService">  
    <intent-filter>  
        <action android:name="com.google.firebase.MESSAGING_EVENT" />  
    </intent-filter>  
</service>  
  
<service  
    android:name=".MyFirebaseInstanceIdService">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT" />  
    </intent-filter>  
</service>
```

第230.2节：接收消息

要接收消息，请使用继承 FirebaseMessagingService 的服务并重写 onMessageReceived 方法。

```
public class MyFcmListenerService extends FirebaseMessagingService {  
  
    /**  
     * 当收到消息时调用。  
     *  
     * @param remoteMessage 表示从 Firebase 云消息传递接收到的消息的对象。  
     */  
    @Override  
    public void onMessageReceived(RemoteMessage message) {
```

Chapter 230: Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user reengagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

Section 230.1: Set Up a Firebase Cloud Messaging Client App on Android

- 完成安装和设置部分，将您的应用连接到 Firebase。
This will create the project in Firebase.

- Add the dependency for Firebase Cloud Messaging to your module-level build.gradle file:

```
dependencies {  
    compile 'com.google.firebaseio:firebase-messaging:10.2.1'  
}
```

Now you are ready to work with the FCM in Android.

FCM clients require devices running Android 2.3 or higher that also have the Google Play Store app installed, or an emulator running Android 2.3 with Google APIs.

Edit your AndroidManifest.xml file

```
<service  
    android:name=".MyFirebaseMessagingService">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.MESSAGING_EVENT" />  
    </intent-filter>  
</service>  
  
<service  
    android:name=".MyFirebaseInstanceIdService">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT" />  
    </intent-filter>  
</service>
```

Section 230.2: Receive Messages

To receive messages, use a service that extends FirebaseMessagingService and override the onMessageReceived method.

```
public class MyFcmListenerService extends FirebaseMessagingService {  
  
    /**  
     * Called when message is received.  
     *  
     * @param remoteMessage Object representing the message received from Firebase Cloud Messaging.  
     */  
    @Override  
    public void onMessageReceived(RemoteMessage message) {
```

```

String from = message.getFrom();

// 检查消息是否包含数据负载。
if (remoteMessage.getData().size() > 0) {
    Log.d(TAG, "消息数据负载: " + remoteMessage.getData());
    Map<String, String> data = message.getData();
}

// 检查消息是否包含通知负载。
if (remoteMessage.getNotification() != null) {
    Log.d(TAG, "消息通知内容: " + remoteMessage.getNotification().getBody());
}

//.....
}

```

当应用处于后台时，Android 会将通知消息发送到系统托盘。用户点击通知时，默认会打开应用启动器。

这包括同时包含通知和数据负载的消息（以及所有从通知控制台发送的消息）。在这些情况下，通知会被发送到设备的系统托盘，数据负载则会作为启动器 Activity 意图的附加信息传递。

这里简要回顾：

应用状态	通知	数据	两者
前台	onMessageReceived	onMessageReceived	onMessageReceived
后台 系统托盘		onMessageReceived 通知：系统托盘 数据：在意图的附加信息中。	

第230.3节：这是我在应用中实现的代码，用于推送图片、消息以及打开 webView 的链接

这是我的FirebaseMessagingService

```

public class MyFirebaseMessagingService extends FirebaseMessagingService {
    Bitmap bitmap;
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String message = remoteMessage.getData().get("message");
        //imageUri将包含要与通知一起显示的图片的URL
        String imageUri = remoteMessage.getData().get("image");
        String link=remoteMessage.getData().get("link");

        //从接收到的URL获取Bitmap图像
        bitmap = getBitmapfromUrl(imageUri);
        sendNotification(message, bitmap,link);

    }

    /**
     * 创建并显示包含接收到的FCM消息的简单通知。
     */

    private void sendNotification(String messageBody, Bitmap image, String link) {
        Intent intent = new Intent(this, NewsListActivity.class);

```

```

String from = message.getFrom();

// Check if message contains a data payload.
if (remoteMessage.getData().size() > 0) {
    Log.d(TAG, "Message data payload: " + remoteMessage.getData());
    Map<String, String> data = message.getData();
}

// Check if message contains a notification payload.
if (remoteMessage.getNotification() != null) {
    Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
}

//.....
}

```

When the app is in the background, Android directs notification messages to the system tray. A user tap on the notification opens the app launcher by default.

This includes messages that contain both notification and data payload (and all messages sent from the Notifications console). In these cases, the notification is delivered to the device's system tray, and the data payload is delivered in the extras of the intent of your launcher Activity.

Here a short recap:

App state	Notification	Data	Both
Foreground	onMessageReceived	onMessageReceived	onMessageReceived
Background	System tray	onMessageReceived	Notification: system tray Data: in extras of the intent.

Section 230.3: This code that i have implemented in my app for pushing image,message and also link for opening in your webView

This is my FirebaseMessagingService

```

public class MyFirebaseMessagingService extends FirebaseMessagingService {
    Bitmap bitmap;
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String message = remoteMessage.getData().get("message");
        //imageUri will contain URL of the image to be displayed with Notification
        String imageUri = remoteMessage.getData().get("image");
        String link=remoteMessage.getData().get("link");

        //To get a Bitmap image from the URL received
        bitmap = getBitmapfromUrl(imageUri);
        sendNotification(message, bitmap,link);

    }

    /**
     * Create and show a simple notification containing the received FCM message.
     */

    private void sendNotification(String messageBody, Bitmap image, String link) {
        Intent intent = new Intent(this, NewsListActivity.class);

```

```

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("LINK", link);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
    PendingIntent.FLAG_ONE_SHOT);
Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
    .setLargeIcon(image)/*通知图标图片*/
.setSmallIcon(R.drawable.hindi)
.setContentTitle(messageBody)
.setStyle(new NotificationCompat.BigPictureStyle()
    .bigPicture(image))/*带图片的通知*/
.setAutoCancel(true)
.setSound(defaultSoundUri)
.setContentIntent(pendingIntent);
NotificationManager notificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(0 /*通知ID*/, notificationBuilder.build());
}

public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;
    } catch (Exception e) {
        // TODO Auto-generated catch block
e.printStackTrace();
        return null;
    }
}
}

```

这是 MainActivity，用于通过意图根据需求在我的 WebView 或其他浏览器中打开链接。

```

if (getIntent().getExtras() != null) {
    if (getIntent().getStringExtra("LINK")!=null) {
        Intent i=new Intent(this,BrowserActivity.class);
i.putExtra("link",getIntent().getStringExtra("LINK"));
i.putExtra("PUSH", "yes");
        NewsListActivity.this.startActivity(i);
        finish();
    }
}

```

第230.4节：注册令牌

在应用程序首次启动时，FCM SDK 会为客户端应用实例生成一个注册令牌。

如果您想针对单个设备或创建设备组，您需要通过扩展 FirebaseInstanceIdService 来访问此令牌。

onTokenRefresh 回调会在生成新令牌时触发，您可以使用方法 FirebaseInstanceId.getToken() 来获取当前令牌。

示例：

```

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("LINK", link);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
    PendingIntent.FLAG_ONE_SHOT);
Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
    .setLargeIcon(image)/*Notification icon image*/
.setSmallIcon(R.drawable.hindi)
.setContentTitle(messageBody)
.setStyle(new NotificationCompat.BigPictureStyle()
    .bigPicture(image))/*Notification with Image*/
.setAutoCancel(true)
.setSound(defaultSoundUri)
.setContentIntent(pendingIntent);
NotificationManager notificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
}

public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;
    } catch (Exception e) {
        // TODO Auto-generated catch block
e.printStackTrace();
        return null;
    }
}
}

```

And this is MainActivity to open link in my WebView or other browser depend on your requirement through intents.

```

if (getIntent().getExtras() != null) {
    if (getIntent().getStringExtra("LINK")!=null) {
        Intent i=new Intent(this,BrowserActivity.class);
i.putExtra("link",getIntent().getStringExtra("LINK"));
i.putExtra("PUSH", "yes");
        NewsListActivity.this.startActivity(i);
        finish();
    }
}

```

Section 230.4: Registration token

On initial startup of your app, the FCM SDK generates a registration token for the client app instance. If you want to target single devices or create device groups, you'll need to access this token by extending FirebaseInstanceIdService.

The onTokenRefresh callback fires whenever a new token is generated and you can use the method FirebaseInstanceId.getToken() to retrieve the current token.

Example:

```

public class MyFirebaseInstanceIdService extends FirebaseInstanceIdService {

    /**
     * 当 InstanceID 令牌更新时调用。如果之前的令牌安全性受到威胁，可能会发生此情况。请注意，这
     * 在 InstanceID 令牌      * 初次生成时也会调用，因此您可以在这里获取令牌。
     */

    @Override
    public void onTokenRefresh() {
        // 获取更新后的InstanceId令牌。
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d(TAG, "已刷新令牌: " + refreshedToken);
    }
}

```

第230.5节：订阅主题

客户端应用可以订阅任何现有主题，或者创建一个新主题。当客户端应用订阅一个新主题名称时，FCM 中会创建一个该名称的新主题，之后任何客户端都可以订阅它。

要订阅主题，请使用 `subscribeToTopic()`方法并指定主题名称：

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

```

public class MyFirebaseInstanceIdService extends FirebaseInstanceIdService {

    /**
     * Called if InstanceID token is updated. This may occur if the security of
     * the previous token had been compromised. Note that this is called when the InstanceID token
     * is initially generated so this is where you would retrieve the token.
     */

    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d(TAG, "Refreshed token: " + refreshedToken);
    }
}

```

Section 230.5: Subscribe to a topic

Client apps can subscribe to any existing topic, or they can create a new topic. When a client app subscribes to a new topic name, a new topic of that name is created in FCM and any client can subsequently subscribe to it.

To subscribe to a topic use the `subscribeToTopic()` method specifying the topic name:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

第231章：Firebase 实时数据库

第231.1节：快速设置

- 完成安装和设置部分，将您的应用连接到 Firebase。
这将会在 Firebase 中创建项目。
- 在模块级别的build.gradle文件中添加 Firebase 实时数据库的依赖：

```
compile 'com.google.firebaseio:firebase-database:10.2.1'
```

- 配置 Firebase 数据库规则

现在你已经准备好在 Android 中使用实时数据库了。

例如，您在数据库中以message键写入一条Hello World消息。

```
// 向数据库写入消息
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

第231.2节：Firebase实时数据库事件处理器

首先初始化FirebaseDatabase：

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

写入数据库：

```
// 向数据库写入消息
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

从数据库读取：

```
// 从数据库读取
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // 此方法在初始值时调用一次,
        // 并且每当该位置的数据更新时再次调用。
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "值为: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // 读取值失败
        Log.w(TAG, "读取值失败。", error.toException());
    }
});
```

Chapter 231: Firebase Realtime DataBase

Section 231.1: Quick setup

- Complete the Installation and setup part to connect your app to Firebase.
This will create the project in Firebase.
- Add the dependency for Firebase Realtime Database to your module-level build.gradle file:

```
compile 'com.google.firebaseio:firebase-database:10.2.1'
```

- Configure Firebase Database Rules

Now you are ready to work with the Realtime Database in Android.

For example you write a Hello World message to the database under the message key.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Section 231.2: Firebase Realtime DataBase event handler

First Initialize FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Write to your database:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Read from your database:

```
// Read from the database
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```

在 Android 事件中检索数据：

```
ChildEventListener childEventListener = new ChildEventListener() {  
    @Override  
    public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildAdded:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildChanged:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildRemoved(DataSnapshot dataSnapshot) {  
        Log.d(TAG, "onChildRemoved:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildMoved:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onCancelled(DatabaseError databaseError) {  
        Log.w(TAG, "postComments:onCancelled", databaseError.toException());  
        Toast.makeText(mContext, "加载评论失败。",  
        Toast.LENGTH_SHORT).show();  
    }  
};  
ref.addChildEventListener(childEventListener);
```

第231.3节：理解firebase JSON数据库

在动手写代码之前，我觉得有必要了解firebase中数据是如何存储的。与关系型数据库不同，firebase以JSON格式存储数据。可以把关系型数据库中的每一行看作一个JSON对象（本质上是无序的键值对）。因此，列名变成了键，而该行中某一列存储的值就是对应的值。这样，整行数据就表示为一个JSON对象，而这些对象的列表则代表整个数据库表。我认为这带来的直接好处是，相比传统的关系型数据库管理系统，模式修改变得更加便宜。向JSON中添加几个属性比修改表结构要容易得多。

下面是一个示例JSON，展示数据在firebase中的存储方式：

```
{  
    "user_base" : {  
        "342343" : {  
            "email" : "kaushal.xxxxx@gmail.com",  
            "authToken" : "some string",  
            "name" : "Kaushal",  
            "phone" : "+919916xxxxxx",  
            "serviceProviderId" : "firebase",  
            "signInServiceType" : "google",  
        },  
        "354895" : {  
            "email" : "xxxxx.devil@gmail.com",  
            "authToken" : "some string",  
        },  
    },  
}
```

Retrieve Data on Android events:

```
ChildEventListener childEventListener = new ChildEventListener() {  
    @Override  
    public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildAdded:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildChanged:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildRemoved(DataSnapshot dataSnapshot) {  
        Log.d(TAG, "onChildRemoved:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {  
        Log.d(TAG, "onChildMoved:" + dataSnapshot.getKey());  
    }  
  
    @Override  
    public void onCancelled(DatabaseError databaseError) {  
        Log.w(TAG, "postComments:onCancelled", databaseError.toException());  
        Toast.makeText(mContext, "Failed to load comments.",  
        Toast.LENGTH_SHORT).show();  
    }  
};  
ref.addChildEventListener(childEventListener);
```

Section 231.3: Understanding firebase JSON database

Before we get our hands dirty with code, I feel it is necessary to understand how data is stored in firebase. Unlike relational databases, firebase stores data in JSON format. Think of each row in a relational database as a JSON object (which is basically unordered key-value pair). So the column name becomes key and the value stored in that column for one particular row is the value. This way the entire row is represented as a JSON object and a list of these represent an entire database table. The immediate benefit that I see for this is schema modification becomes much more cheaper operation compared to old RDBMS. It is easier to add a couple of more attributes to a JSON than altering a table structure.

here is a sample JSON to show how data is stored in firebase:

```
{  
    "user_base" : {  
        "342343" : {  
            "email" : "kaushal.xxxxx@gmail.com",  
            "authToken" : "some string",  
            "name" : "Kaushal",  
            "phone" : "+919916xxxxxx",  
            "serviceProviderId" : "firebase",  
            "signInServiceType" : "google",  
        },  
        "354895" : {  
            "email" : "xxxxx.devil@gmail.com",  
            "authToken" : "some string",  
        },  
    },  
}
```

```

    "name" : "魔鬼",
    "phone" : "+919685xxxxx",
    "serviceProviderId" : "firebase",
    "signInServiceType" : "github"
},
"371298" : {
    "email" : "bruce.wayne@wayneinc.com",
    "authToken" : "我是蝙蝠侠",
    "name" : "布鲁斯·韦恩",
    "phone" : "+14085xxxxx",
    "serviceProviderId" : "firebase",
    "signInServiceType" : "shield"
}
},
"user_prefs" : {
    "key1" :{
        "data": "针对键一"
    },
    "key2":{
        "data" : "for key two"
    },
    "key3":{
        "data" : "for key three"
    }
},
//其他结构
}

```

这清楚地展示了我们过去存储在关系型数据库中的数据如何以JSON格式存储。接下来让我们看看如何在安卓设备中读取这些数据。

第231.4节：从Firebase检索数据

我假设你已经了解如何在Android Studio中添加Firebase的Gradle依赖。如果不了解只需按照 [here](#) 的指南操作。在 Firebase控制台添加你的应用，[添加依赖后](#)在Android Studio中同步Gradle。并非所有依赖都需要，只需Fireb ase数据库和Firebase认证。

既然我们知道了数据如何存储以及如何添加Gradle依赖，接下来看看如何使用导入的 Firebase Android SDK来检索数据。

创建一个Firebase数据库引用

```

DatabaseReference userDBRef = FirebaseDatabase.getInstance().getReference();
// 上述语句指向根节点
userDBRef = DatabaseReference.getInstance().getReference().child("user_base")
// 指向 user_base 表的 JSON (见前一节)

```

从这里你可以链式调用多个 child() 方法来指向你感兴趣的数据。例如，如果数据如前一节所示存储，且你想指向布鲁斯·韦恩 (Bruce Wayne) 用户，可以使用：

```

DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 是 JSON 结构中布鲁斯·韦恩用户的键 (前一节)

```

或者直接传递整个 JSON 对象的引用：

```

DatabaseReference bruceWayneRef = DatabaseReference.getInstance().getReference()
    .child("user_base/371298");
// 深层嵌套的数据也可以通过这种方式引用，只需将完整的

```

```

    "name" : "devil",
    "phone" : "+919685xxxxx",
    "serviceProviderId" : "firebase",
    "signInServiceType" : "github"
},
"371298" : {
    "email" : "bruce.wayne@wayneinc.com",
    "authToken" : "I am batman",
    "name" : "Bruce Wayne",
    "phone" : "+14085xxxxx",
    "serviceProviderId" : "firebase",
    "signInServiceType" : "shield"
}
},
"user_prefs" : {
    "key1" :{
        "data": "for key one"
    },
    "key2":{
        "data" : "for key two"
    },
    "key3":{
        "data" : "for key three"
    }
},
//other structures
}

```

This clearly shows how data that we used to store in relational databases can be stored in JSON format. Next let's see how to read this data in android devices.

Section 231.4: Retrieving data from firebase

I am gonna assume you already know about adding gradle dependencies firebase in android studio. If you don't just follow the guide from [here](#). Add your app in firebase console, gradle sync android studio after adding dependencies. All dependencies are not needed just firebase database and firebase auth.

Now that we know how data is stored and how to add gradle dependencies let's see how to use the imported firebase android SDK to retrieve data.

create a firebase database reference

```

DatabaseReference userDBRef = FirebaseDatabase.getInstance().getReference();
// above statement point to base tree
userDBRef = DatabaseReference.getInstance().getReference().child("user_base")
// points to user_base table JSON (see previous section)

```

from here you can chain multiple child() method calls to point to the data you are interested in. For example if data is stored as depicted in previous section and you want to point to Bruce Wayne user you can use:

```

DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)

```

Or simply pass the whole reference to the JSON object:

```

DatabaseReference bruceWayneRef = DatabaseReference.getInstance().getReference()
    .child("user_base/371298");
// deeply nested data can also be referenced this way, just put the fully

```

```
// 路径按上述代码示例的模式写出 "blah/blah1/blah1-2/blah1-2-3..."
```

既然我们已经有了要获取的数据的参考，我们就可以使用监听器在安卓应用中获取数据。

与传统使用 retrofit 或 volley 发起 REST API 调用不同，这里只需一个简单的回调监听器即可获取数据。Firebase SDK 会调用回调方法，操作完成。

基本上可以附加两种监听器，一种是ValueEventListener，另一种是ChildEventListener（在下一节中描述）。对于我们有引用并添加监听器的节点下数据的任何变化，值事件监听器会返回整个 JSON 结构，而子事件监听器返回发生变化的具体子节点。这两者各有用途。要从 Firebase 获取数据，我们可以向 Firebase 数据库引用（如之前创建的 userDBRef）添加一个或多个监听器。

以下是一些示例代码（代码后有解释）：

```
userDBRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        User bruceWayne = dataSnapshot.child("371298").getValue(User.class);
        // 对检索到的数据或 Bruce Wayne 进行操作
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e("UserListActivity", "发生错误");
        // 对错误进行处理
    }
});
```

你注意到传入的类类型了吗？DataSnapshot 可以将 JSON 数据转换为我们定义的 POJO，只需传入正确的类类型。

如果你的使用场景不需要每次数据（在本例中是 user_base 表）有细微变化时都获取全部数据，或者说你只想获取数据一次，可以使用数据库引用的addListenerForSingleValueEvent()方法。该方法只会触发一次回调。

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // 执行某些操作
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // 处理错误
    }
});
```

以上示例将为您提供JSON节点的值。要获取键，只需调用：

```
String myKey = dataSnapshot.getKey();
```

第231.5节：监听子节点更新

以聊天应用或协作购物清单应用为例（基本上需要一个对象列表在用户之间同步）。如果您使用Firebase数据库并向聊天父节点或购物清单父节点添加值事件监听器，每当添加聊天节点（即有人说“嗨”）时，您将获得从聊天开始以来的整个聊天结构。这不是我们想要的，我们只关心新节点，或者被删除或修改的旧节点，未改变的节点不应返回。

```
// qualified path in pattern shown in above code "blah/blah1/blah1-2/blah1-2-3..."
```

Now that we have the reference of the data we want to fetch, we can use listeners to fetch data in android apps. Unlike the traditional calls where you fire REST API calls using retrofit or volley, here a simple callback listener is required to get the data. Firebase sdk calls the callback methods and you are done.

There are basically two types of listeners you can attach, one is [ValueEventListener](#) and the other one is [ChildEventListener](#) (described in next section). For any change in data under the node we have references and added listeners to, value event listeners return the entire JSON structure and child event listener returns specific child where the change has happened. Both of these are useful in their own way. To fetch the data from firebase we can add one or more listeners to a firebase database reference (list userDBRef we created earlier).

Here is some sample code (code explanation after code):

```
userDBRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        User bruceWayne = dataSnapshot.child("371298").getValue(User.class);
        // Do something with the retrieved data or Bruce Wayne
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e("UserListActivity", "Error occurred");
        // Do something about the error
    }
});
```

Did you notice the Class type passed. [DataSnapshot](#) can convert JSON data into our defined POJOs, simple pass the right class type.

If your use case does not require the entire data (in our case user_base table) every time some little change occurs or say you want to **fetch the data only once**, you can use [addListenerForSingleValueEvent\(\)](#) method of Database reference. This fires the callback only once.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    }
});
```

Above samples will give you the value of the JSON node. To get the key simply call:

```
String myKey = dataSnapshot.getKey();
```

Section 231.5: Listening for child updates

Take a use case, like a chat app or a collaborative grocery list app (that basically requires a list of objects to be synced across users). If you use firebase database and add a value event listener to the chat parent node or grocery list parent node, you will end with entire chat structure from the beginning of time (i meant beginning of your chat) every time a chat node is added (i.e. anyone says hi). That we don't want to do, what we are interested in is only the new node or only the old node that got deleted or modified, the unchanged ones should not be returned.

在这种情况下，我们可以使用`ChildEventListener`。废话不多说，以下是代码示例（请参见前面章节的示例JSON数据）：

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //如果不处理有序数据，可以忽略此方法
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
});
```

方法名不言自明。如你所见，每当添加新用户、修改现有用户的某些属性，或删除/移除用户时，都会调用子事件监听器的相应回调方法，并传入相关数据。因此，如果你想保持界面刷新，比如聊天应用，可以从`onChildAdded()`获取 JSON，解析成 POJO 并填充到界面中。只需记得用户离开界面时移除监听器。

`onChildChanged()` 返回包含更改属性（新增属性）的整个子节点值。

`onChildRemoved()` 返回被移除的子节点。

第231.6节：使用分页检索数据

当你的 JSON 数据库非常庞大时，添加值事件监听器没有意义。它会返回庞大的 JSON，解析起来非常耗时。在这种情况下，我们可以使用分页，分批获取数据并显示或处理。类似于懒加载，或者用户点击“显示更早聊天”时获取旧聊天记录。这时可以使用`Query`。

我们以之前章节的例子为例。用户库包含3个用户，如果增长到30万用户，并且你想分批获取每批50个用户列表：

```
// 类级别
final int limit = 50;
int start = 0;

// 事件级别
Query userListQuery = userDBRef.orderByChild("email").limitToFirst(limit)
    .startAt(start)
userListQuery.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // 执行某些操作
        start += (limit+1);
    }
});
```

In this case we can use `ChildEventListener`. Without any further adieu, here is code sample (see prev sections for sample JSON data):

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //If not dealing with ordered data forget about this
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
});
```

Method names are self explanatory. As you can see whenever a new user is added or some property of existing user is modified or user is deleted or removed appropriate callback method of child event listener is called with relevant data. So if you are keeping UI refreshed for say chat app, get the JSON from `onChildAdded()` parse into POJO and fit it in your UI. Just remember to remove your listener when user leaves the screen.

`onChildChanged()` gives the entire child value with changed properties (new ones).

`onChiledRemoved()` returns the removed child node.

Section 231.6: Retrieving data with pagination

When you have a huge JSON database, adding a value event listener doesn't make sense. It will return the huge JSON and parsing it would be time consuming. In such cases we can use pagination and fetch part of data and display or process it. Kind of like lazy loading or like fetching old chats when user clicks on show older chat. In this case `Query` can used.

Let's take the our old example in previous sections. The user base contains 3 users, if it grows to say 3 hundred thousand user and you want to fetch the user list in batches of 50:

```
// class level
final int limit = 50;
int start = 0;

// event level
Query userListQuery = userDBRef.orderByChild("email").limitToFirst(limit)
    .startAt(start)
userListQuery.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
        start += (limit+1);
    }
});
```

```

@Override
public void onCancelled(DatabaseError databaseError) {
    // 处理错误
});

```

这里可以添加并监听 value 或 child 事件。再次调用查询以获取接下来的 50 条。确保添加了orderByChild() 方法，否则无法正常工作。Firebase 需要知道你分页的排序依据。

第 231.7 节：反规范化：扁平数据库结构

反规范化和扁平数据库结构对于高效下载分页调用是必要的。使用以下结构，也可以维护双向关系。这种方法的缺点是，你总是需要在多个地方更新数据。

举个例子，想象一个允许用户给自己存储消息（备忘录）的应用。

期望的扁平数据库结构：

```

|--database
|---备忘录
|   |--备忘录键1
|       |--标题: "标题"
|       |--内容: "信息"
|   |--备忘录键2
|       |--标题: "重要标题"
|       |--内容: "重要信息"
|---用户
|   |--用户键1
|       |--姓名: "约翰·多伊"
|       |--备忘录
|           |-备忘录键1 : true //这里的值无关紧要，我们只需要键。
|           |--备忘录键2 : true
|   |--用户键2
|       |--姓名: "马克斯·多伊"

```

使用的备忘录类

```

public class Memo {
    private String title, content;
    // 获取器和设置器 ...

    // toMap() 对推送过程是必要的
    private Map<String, Object> toMap() {
        HashMap<String, Object> result = new HashMap<>();
        result.put("title", title);
        result.put("content", content);
        return result;
    }
}

```

检索用户的备忘录

```

//我们需要分别存储键和备忘录
private ArrayList<String> mKeys = new ArrayList<>();
private ArrayList<Memo> mMemos = new ArrayList<>();

//用户需要登录以检索 uid

```

```

@Override
public void onCancelled(DatabaseError databaseError) {
    // Do something about the error
});

```

Here value or child events can be added and listened to. Call query again to fetch next 50. **Make sure to add the orderByChild() method**, this will not work without that. Firebase needs to know the order by which you are paginating.

Section 231.7: Denormalization: Flat Database Structure

Denormalization and a flat database structure is necessary to efficiently download separate calls. With the following structure, it is also possible to maintain two-way relationships. The disadvantage of this approach is, that you always need to update the data in multiple places.

For an example, imagine an app which allows the user to store messages to himself (memos).

Desired flat database structure:

```

|--database
|--- memos
|   |-- memokey1
|       |-- title: "Title"
|       |-- content: "Message"
|   |-- memokey2
|       |-- title: "Important Title"
|       |-- content: "Important Message"
|--- users
|   |-- userKey1
|       |-- name: "John Doe"
|       |-- memos
|           |-- memokey1 : true //The values here don't matter, we only need the keys.
|           |-- memokey2 : true
|   |-- userKey2
|       |-- name: "Max Doe"

```

The used memo class

```

public class Memo {
    private String title, content;
    // getters and setters ...

    // toMap() is necessary for the push process
    private Map<String, Object> toMap() {
        HashMap<String, Object> result = new HashMap<>();
        result.put("title", title);
        result.put("content", content);
        return result;
    }
}

```

Retrieving the memos of a user

```

//We need to store the keys and the memos separately
private ArrayList<String> mKeys = new ArrayList<>();
private ArrayList<Memo> mMemos = new ArrayList<>();

//The user needs to be logged in to retrieve the uid

```

```

String currentUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();

//这是对用户备忘录列表的引用
DatabaseReference currentUserMemoReference = FirebaseDatabase.getInstance().getReference()
    .child("users").child(currentUserId).child("memos");

//这是指向所有备忘录列表的引用
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference()
    .child("memos");

//我们开始监听用户的备忘录,
//这也会初始检索备忘录
currentUserMemoReference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        //这里我们获取用户备忘录的键值。
        String key = dataSnapshot.getKey(); //例如 memokey1
        //为了后续操作列表,我们需要将键存储在列表中
        mKeys.add(key);
        //既然知道了哪个消息属于用户,
        //我们就从备忘录中请求它：
        memoReference.child(key).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //这里我们获取备忘录内容：
                Memo memo = dataSnapshot.getValue(Memo.class);
                mMemos.add(memo);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) { }
        });
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {}

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}

    @Override
    public void onCancelled(DatabaseError databaseError) { }
})
}

```

创建备忘录

```

//用户需要登录才能获取 uid
String currentUserUid = FirebaseAuth.getInstance().getCurrentUser().getUid();

//这是用户备忘录列表的路径
String userMemoPath = "users/" + currentUserUid + "/memos/";

//这是所有备忘录列表的路径
String memoPath = "memos/";

//我们需要从备忘录引用中获取一个未使用的键
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference().child("memos");
String key = memoReference.push().getKey();

```

```

String currentUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the reference to the list of memos a user has
DatabaseReference currentUserMemoReference = FirebaseDatabase.getInstance().getReference()
    .child("users").child(currentUserId).child("memos");

//This is a reference to the list of all memos
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference()
    .child("memos");

//We start to listen to the users memos,
//this will also retrieve the memos initially
currentUserMemoReference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        //Here we retrieve the key of the memo the user has.
        String key = dataSnapshot.getKey(); //for example memokey1
        //For later manipulations of the lists, we need to store the key in a list
        mKeys.add(key);
        //Now that we know which message belongs to the user,
        //we request it from our memos:
        memoReference.child(key).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //Here we retrieve our memo:
                Memo memo = dataSnapshot.getValue(Memo.class);
                mMemos.add(memo);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) { }
        });
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {}

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}

    @Override
    public void onCancelled(DatabaseError databaseError) { }
})
}

```

Creating a memo

```

//The user needs to be logged in to retrieve the uid
String currentUserUid = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the path to the list of memos a user has
String userMemoPath = "users/" + currentUserUid + "/memos/";

//This is the path to the list of all memos
String memoPath = "memos/";

//We need to retrieve an unused key from the memos reference
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference().child("memos");
String key = memoReference.push().getKey();

```

```

Memo newMemo = new Memo("重要数字", "1337, 42, 3.14159265359");

Map<String, Object> childUpdates = new HashMap<>();
//这里的第二个参数(值)无关紧要,只要键存在即可
childUpdates.put(userMemoPath + key, true);
childUpdates.put(memoPath + key, newMemo.toMap());

FirebaseDatabase.getInstance().getReference().updateChildren(childUpdates);

```

推送后,数据库看起来像这样:

```

|--database
|---备忘录
|   |--备忘录键1
|       |--标题: "标题"
|       |--内容: "信息"
|   |--备忘录键2
|       |-- title: "重要标题"
|       |-- content: "重要信息"
|   |-- generatedMemekey3
|       |-- 标题: "重要数字"
|       |-- 内容: "1337, 42, 3.14159265359"
|--- 用户
|   |--用户键1
|       |-- 姓名: "约翰·多伊"
|   |--备忘录
|       /--备忘录键1 : true //这里的值无关紧要,我们只需要键。
|           |-- memokey2 : true
|           |-- generatedMemekey3 : true
|   |-- userKey2
|       |-- 姓名: "马克斯·多伊"

```

第231.8节：设计和理解如何从Firebase数据库中检索实时数据

本示例假设您已经设置好了Firebase实时数据库。如果您是初学者,请在此处了解如何将Firebase添加到您的Android项目中。

首先,将Firebase数据库的依赖项添加到应用级别的build.gradle文件中:

```
compile 'com.google.firebaseio:firebase-database:9.4.0'
```

现在,让我们创建一个将数据存储到Firebase数据库中的聊天应用。

步骤1: 创建一个名为Chat的类

只需创建一个包含聊天所需基本变量的类:

```
public class Chat{
    public String name, message;
}
```

步骤2: 创建一些JSON数据

为了向 Firebase 数据库发送/检索数据,您需要使用 JSON。假设数据库根级别已经存储了一些聊天记录。这些聊天记录的数据可能如下所示:

[

```

Memo newMemo = new Memo("Important numbers", "1337, 42, 3.14159265359");

Map<String, Object> childUpdates = new HashMap<>();
//The second parameter **here** (the value) does not matter, it's just that the key exists
childUpdates.put(userMemoPath + key, true);
childUpdates.put(memoPath + key, newMemo.toMap());

FirebaseDatabase.getInstance().getReference().updateChildren(childUpdates);

```

After the push, or database looks like this:

```

|--database
|--- memos
|   |-- memokey1
|       |-- title: "Title"
|       |-- content: "Message"
|   |-- memokey2
|       |-- title: "Important Title"
|       |-- content: "Important Message"
|   |-- generatedMemekey3
|       |-- title: "Important numbers"
|       |-- content: "1337, 42, 3.14159265359"
|--- users
|   |-- userKey1
|       |-- name: "John Doe"
|       |-- memos
|           |-- memokey1 : true //The values here don't matter, we only need the keys.
|           |-- memokey2 : true
|           |-- generatedMemekey3 : true
|   |-- userKey2
|       |-- name: "Max Doe"

```

Section 231.8: Designing and understanding how to retrieve realtime data from the Firebase Database

This example assumes that you have already set up a Firebase Realtime Database. If you are a starter, then please inform yourself [here](#) on how to add Firebase to your Android project.

First, add the dependency of the Firebase Database to the app level build.gradle file:

```
compile 'com.google.firebaseio:firebase-database:9.4.0'
```

Now, let us create a chat app which stores data into the Firebase Database.

Step 1: Create a class named Chat

Just create a class with some basic variables required for the chat:

```
public class Chat{
    public String name, message;
}
```

Step 2: Create some JSON data

For sending/retrieving data to/from the Firebase Database, you need to use JSON. Let us assume that some chats are already stored at the root level in the database. The data of these chats could look like as follows:

[

```

{
    "name": "约翰·多伊",
    "message": "我的第一条消息"
},
{
    "name": "约翰·多伊",
    "message": "第二条消息"
},
{
    "name": "约翰·多伊",
    "message": "第三条消息"
}
]

```

步骤 3：添加监听器

监听器有三种类型。在下面的示例中，我们将使用childEventListener：

```

DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference() // 引用数据库的根节点。
    .child("chats"); // 引用根节点下的 "chats" 节点。

chatDb.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        // 该函数会为每个子节点调用，在此例中，即每个聊天节点都会调用，所以根据上述示例，该函数将被调用3次。
        // 从该函数中获取 Chat 对象很简单。
        Chat chat; // 创建一个空的 Chat 对象。

        // 使用 dataSnapshot 的 getValue 函数，并传入你想转换成的对象的类名，// 在此例中是 Chat.class。
        chat = dataSnapshot.getValue(Chat.class);

        // 现在你可以使用这个 chat 对象，将其添加到 ArrayList 或类似的集合中，// 并在 RecyclerView 中显示。
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        // 当任何节点的值发生变化时会调用该函数，dataSnapshot 会获取带有子节点键的数据，// 你可以用新值替换 ArrayList 中的旧值或类似操作。
        // 要获取密钥，请使用 .getKey() 函数。
        // 要获取值，请使用类似上述的代码。
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        // 当任何子节点被移除时调用此函数。dataSnapshot 将// 获取该子节点的键对应的数据。
        // 要获取键，请使用 s 字符串参数。
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        // 当任何子节点被移动到不同位置时调用此函数。
        // 要获取键，请使用 s 字符串参数。
    }
})

```

```

{
    "name": "John Doe",
    "message": "My first Message"
},
{
    "name": "John Doe",
    "message": "Second Message"
},
{
    "name": "John Doe",
    "message": "Third Message"
}
]

```

Step 3: Adding the listeners

There are three types of listeners. In the following example we are going to use the childEventListener:

```

DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference() // Referencing the root of the database.
    .child("chats"); // Referencing the "chats" node under the root.

chatDb.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        // This function is called for every child id chat in this case, so using the above example, this function is going to be called 3 times.

        // Retrieving the Chat object from this function is simple.
        Chat chat; // Create a null chat object.

        // Use the getValue function in the dataSnapshot and pass the object's class name to // which you want to convert and get data. In this case it is Chat.class.
        chat = dataSnapshot.getValue(Chat.class);

        // Now you can use this chat object and add it into an ArrayList or something like // that and show it in the recycler view.
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the node value is changed, dataSnapshot will // get the data with the key of the child, so you can swap the new value with the // old one in the ArrayList or something like that.

        // To get the key, use the .getKey() function.
        // To get the value, use code similar to the above one.
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        // This function is called when any of the child node is removed. dataSnapshot will // get the data with the key of the child.

        // To get the key, use the s String parameter .
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the child nodes is moved to a different position.

        // To get the key, use the s String parameter.
    }
})

```

```
}

@Override
public void onCancelled(DatabaseError databaseError) {
    // 如果出现任何错误，将调用此函数。

    // 你可以通过 databaseError.toException() 获取异常信息；
}

});
```

步骤4：向数据库添加数据

只需创建一个 Chat 类对象，并按如下方式添加值：

```
Chat chat=new Chat();
chat.name="约翰·多伊";
chat.message="来自安卓的第一条消息";
```

现在获取对 chats 节点的引用，方法与检索会话时相同：

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

在开始添加数据之前，请记住你还需要一个更深层的引用，因为一个聊天节点包含多个子节点，添加新聊天意味着添加一个包含聊天详情的新节点。我们可以使用 push() 函数在 DatabaseReference 对象上生成一个新的唯一节点名称，该函数会返回另一个 DatabaseReference，指向一个新创建的节点，用于插入聊天数据。

示例

```
// 参数是上面几行新创建的聊天对象。
chatDb.push().setValue(chat);
```

setValue() 函数会确保所有应用的 onDataChange 函数被调用（包括同一设备上的），这些函数是“chats”节点的附加监听器。

```
}

@Override
public void onCancelled(DatabaseError databaseError) {
    // If anything goes wrong, this function is going to be called.

    // You can get the exception by using databaseError.toException();
}

});
```

Step 4: Add data to the database

Just create a Chat class object and add the values as follows:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Now get a reference to the chats node as done in the retrieving session:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Before you start adding data, keep in mind that you need one more deep reference since a chat node has several more nodes and adding a new chat means adding a new node containing the chat details. We can generate a new and unique name of the node using the push() function on the DatabaseReference object, which will return another DatabaseReference, which in turn points to a newly formed node to insert the chat data.

Example

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

The setValue() function will make sure that all of the application's onDataChange functions are getting called (including the same device), which happens to be the attached listener of the "chats" node.

第232章：Firebase 应用索引

第232.1节：支持Http URL

步骤1：允许Google抓取您的内容。编辑服务器的robot.txt文件。您可以通过编辑此文件来控制Google对您内容的抓取，详情请参考此链接。

步骤2：将您的应用与您的网站关联。包含assetlinks.json文件，您需要将其上传到您的网站服务器的.well-known目录。assetlinks.json的内容如下：

```
[  
  {  
    "relation": ["delegate_permission/common.handle_all_urls"],  
    "target":  
      { "namespace": "android_app",  
        "package_name": "<your_package_name>",  
        "sha256_cert_fingerprints": ["<hash_of_app_certificate>"] }  
  }]
```

步骤3：在您的清单文件中包含应用链接，以便将URL重定向到您的应用，示例如下：

```
<activity  
    android:name=".activity.SampleActivity"  
    android:label="@string/app_name"  
    android:windowSoftInputMode="adjustResize|stateAlwaysHidden">  
    <intent-filter>  
        <action android:name="android.intent.action.VIEW" />  
        <category android:name="android.intent.category.DEFAULT" />  
        <category android:name="android.intent.category.BROWSABLE" />  
    <data  
        android:host="example.live"  
        android:pathPrefix="/vod"  
        android:scheme="https"/>  
    <data  
        android:host="example.live"  
        android:pathPrefix="/vod"  
        android:scheme="http"/>  
    </intent-filter>  
</activity>
```

如果您想了解这里的每一个标签，请参阅此处。

< action> 指定 ACTION_VIEW 意图动作，以便意图过滤器可以从 Google 搜索中访问。

< data> 添加一个或多个标签，每个标签代表一个解析到该活动的 URI 格式。至少，标签必须包含 android:scheme 属性。你可以添加额外的属性来进一步细化活动接受的 URI 类型。例如，你可能有多个接受类似 URI 的活动，但仅基于路径名称不同。在这种情况下，使用 android:path 属性或其变体 (pathPattern 或 pathPrefix) 来区分系统应为不同 URI 路径打开哪个活动。

< category> 包含 BROWSABLE 类别。BROWSABLE 类别是使意图过滤器可从网页浏览器访问所必需的。没有它，点击浏览器中的链接无法解析到你的应用。DEFAULT 类别是可选的，但推荐使用。没有此类别，活动只能通过显式意图使用你的应用组件名称启动。

步骤 4：处理传入的 URL

Chapter 232: Firebase App Indexing

Section 232.1: Supporting Http URLs

Step 1: Allow Google to Crawl to your content.Edit server's robot.txt file.You can control google crawling for your content by editing this file,you can refer to [this link](#) for more details.

Step 2: Associate your App with your website.Include assetlinks.json You upload it to your web server's .well-known directory.Content of your assetlinks.json are as-

```
[  
  {  
    "relation": ["delegate_permission/common.handle_all_urls"],  
    "target":  
      { "namespace": "android_app",  
        "package_name": "<your_package_name>",  
        "sha256_cert_fingerprints": ["<hash_of_app_certificate>"] }  
  }]
```

Step 3: Include App links in your manifest file to redirectUrls into your Application like below,

```
<activity  
    android:name=".activity.SampleActivity"  
    android:label="@string/app_name"  
    android:windowSoftInputMode="adjustResize|stateAlwaysHidden">  
    <intent-filter>  
        <action android:name="android.intent.action.VIEW" />  
        <category android:name="android.intent.category.DEFAULT" />  
        <category android:name="android.intent.category.BROWSABLE" />  
    <data  
        android:host="example.live"  
        android:pathPrefix="/vod"  
        android:scheme="https"/>  
    <data  
        android:host="example.live"  
        android:pathPrefix="/vod"  
        android:scheme="http"/>  
    </intent-filter>  
</activity>
```

Refer to this if you want learn about each and every tag here.

< action> Specify the ACTION_VIEW intent action so that the intent filter can be reached from Google Search.

< data> Add one or more tags, where each tag represents a URI format that resolves to the activity. At minimum, the tag must include the android:scheme attribute. You can add additional attributes to further refine the type of URI that the activity accepts. For example, you might have multiple activities that accept similar URLs, but which differ simply based on the path name. In this case, use the android:path attribute or its variants (pathPattern or pathPrefix) to differentiate which activity the system should open for different URI paths.

< category> Include the BROWSABLE category. The BROWSABLE category is required in order for the intent filter to be accessible from a web browser. Without it, clicking a link in a browser cannot resolve to your app. The DEFAULT category is optional, but recommended. Without this category, the activity can be started only with an explicit intent, using your app component name.

Step 4: Handle incoming URLs

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_schedule);
    onNewIntent(getIntent());
}

protected void onNewIntent(Intent intent) {
    String action = intent.getAction();
    Uri data = intent.getData();
    if (Intent.ACTION_VIEW.equals(action) && data != null) {
        articleId = data.getLastPathSegment();
        TextView linkText = (TextView) findViewById(R.id.link);
        linkText.setText(data.toString());
    }
}

```

步骤5：您可以通过使用Android调试桥命令或Android Studio配置来测试。Adb命令：启动您的应用程序，然后运行此命令：

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Android Studio配置：Android Studio > 构建 > 编辑配置 > 启动选项 > 选择URL > 然后在此处输入您的URL > 应用并测试。如果“运行”窗口显示错误，则需要检查您的URL格式是否与清单文件中提到的applinks匹配，否则将成功运行，并重定向到您指定的URL页面。

第232.2节：添加AppIndexing API

关于将此添加到项目，您可以轻松找到官方文档，但在此示例中，我将重点介绍一些需要注意的关键点。

步骤1：添加Google服务

```

dependencies {
    ...
    compile 'com.google.android.gms:play-services-appindexing:9.4.0'
    ...
}

```

步骤2：导入类

```

import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;

```

步骤3：添加应用索引API调用

```

private GoogleApiClient mClient;
private Uri mUrl;
private String mTitle;
private String mDescription;

//如果你知道要索引的值，可以在onCreate()中初始化这些变量
@Override
protected void onCreate(Bundle savedInstanceState) {

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_schedule);
    onNewIntent(getIntent());
}

protected void onNewIntent(Intent intent) {
    String action = intent.getAction();
    Uri data = intent.getData();
    if (Intent.ACTION_VIEW.equals(action) && data != null) {
        articleId = data.getLastPathSegment();
        TextView linkText = (TextView) findViewById(R.id.link);
        linkText.setText(data.toString());
    }
}

```

Step 5: You can test this by using Android Debug Bridge command or studio configurations. Adb command: Launch your application and then run this command:

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Android Studio Configurations: **Android studio > Build > Edit Configuration >Launch options>select URL>then type in your Url here >Apply** and test.Run your application if “Run” window shows error then you need to check your URL format with your applinks mentioned in manifest otherwise it will successfully run, and redirect to page mentioned your URL if specified.

Section 232.2: Add AppIndexing API

For Adding this to project you can find official doc easily but in this example I'm going to highlight some of the key areas to be taken care of.

Step 1: Add google service

```

dependencies {
    ...
    compile 'com.google.android.gms:play-services-appindexing:9.4.0'
    ...
}

```

Step 2: Import classes

```

import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;

```

Step 3: Add App Indexing API calls

```

private GoogleApiClient mClient;
private Uri mUrl;
private String mTitle;
private String mDescription;

//If you know the values that to be indexed then you can initialize these variables in onCreate()
@Override
protected void onCreate(Bundle savedInstanceState) {

```

```

mClient = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
mUrl = "http://examplepetstore.com/dogs/standard-poodle";
mTitle = "标准贵宾犬";
mDescription = "标准贵宾犬肩高至少18英寸";
}

//如果你的数据来自网络请求，则在 onResponse() 中初始化这些值，并检查空指针异常 (NPE)，以防代码崩溃。
//设置应用索引的标题和描述
mUrl = Uri.parse("android-app://com.famelive/https/m.fame.live/vod/" +model.getId());
mTitle = model.getTitle();
mDescription = model.getDescription();

mClient.connect();
AppIndex.AppIndexApi.start(mClient, getAction());

@Override
protected void onStop() {
if (mTitle != null && mDescription != null && mUrl != null) //如果响应失败，请检查这些是否已初始化
    if (getAction() != null) {
        AppIndex.AppIndexApi.end(mClient, getAction());
        mClient.disconnect();
    }
    super.onStop();
}

public Action getAction() {
    Thing object = new Thing.Builder()
        .setName(mTitle)
.setDescription(mDescription)
        .setUrl(mUrl)
.build();

    return new Action.Builder(Action.TYPE_WATCH)
        .setObject(object)
.setActionStatus(Action.STATUS_TYPE_COMPLETED)
        .build();
}

```

要测试此功能，只需按照下面备注中的第4步操作。

```

mClient = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
mUrl = "http://examplepetstore.com/dogs/standard-poodle";
mTitle = "Standard Poodle";
mDescription = "The Standard Poodle stands at least 18 inches at the withers";
}

//If your data is coming from a network request, then initialize these value in onResponse() and make
checks for NPE so that your code won't fall apart.

//setting title and description for App Indexing
mUrl = Uri.parse("android-app://com.famelive/https/m.fame.live/vod/" +model.getId());
mTitle = model.getTitle();
mDescription = model.getDescription();

mClient.connect();
AppIndex.AppIndexApi.start(mClient, getAction());

@Override
protected void onStop() {
if (mTitle != null && mDescription != null && mUrl != null) //if your response fails then check
whether these are initialized or not
    if (getAction() != null) {
        AppIndex.AppIndexApi.end(mClient, getAction());
        mClient.disconnect();
    }
    super.onStop();
}

public Action getAction() {
    Thing object = new Thing.Builder()
        .setName(mTitle)
        .setDescription(mDescription)
        .setUrl(mUrl)
        .build();

    return new Action.Builder(Action.TYPE_WATCH)
        .setObject(object)
        .setActionStatus(Action.STATUS_TYPE_COMPLETED)
        .build();
}

```

To test this just follow the step 4 in Remarks given below.

第233章：Firebase崩溃报告

第233.1节：如何报告错误

Firebase崩溃报告会自动生成致命错误（或未捕获异常）的报告。

您可以使用以下方式创建自定义报告：

```
FirebaseCrash.report(new Exception("我的第一个Android非致命错误"));
```

您可以在日志中查看FirebaseCrash初始化模块的时间：

```
07-20 08:57:24.442 D/FirebaseCrashApiImpl: FirebaseCrash报告API已初始化 07-20  
08:57:24.442 I/FirebaseCrash: FirebaseCrash报告初始化完成  
com.google.firebaseio.crash.internal.zzz@3333d325 07-20 08:57:24.442 D/FirebaseApp: 已初始化类  
com.google.firebaseio.crash.FirebaseCrash.
```

然后在发送异常时：

```
07-20 08:57:47.052 D/FirebaseCrashApiImpl: throwable java.lang.Exception: 我的第一个Android非  
致命错误 07-20 08:58:18.822 D/FirebaseCrashSenderServiceImpl: 响应代码：200 07-20  
08:58:18.822 D/FirebaseCrashSenderServiceImpl: 报告已发送
```

您可以通过以下方式向报告添加自定义日志

```
FirebaseCrash.log("活动已创建");
```

第233.2节：如何将Firebase Crash Reporting添加到您的

应用程序

为了将Firebase Crash Reporting添加到您的应用程序，请执行以下步骤：

- 在Firebase 控制台创建一个应用程序。
- 将项目中的google-services.json文件复制到你的app/目录中。
- 在根目录的build.gradle文件中添加以下规则，以包含google-services插件：

```
buildscript {  
    // ...  
    dependencies {  
        // ...  
        classpath 'com.google.gms:google-services:3.0.0'  
    }  
}  
// ...
```

- 在你的模块Gradle文件底部添加apply plugin行以启用Gradle插件：

```
apply plugin: 'com.google.gms.google-services'
```

Chapter 233: Firebase Crash Reporting

Section 233.1: How to report an error

Firebase Crash Reporting automatically generates reports for fatal errors (or uncaught exceptions).

You can create your custom report using:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

You can check in the log when FirebaseCrash initialized the module:

```
07-20 08:57:24.442 D/FirebaseCrashApiImpl: FirebaseCrash reporting API initialized 07-20  
08:57:24.442 I/FirebaseCrash: FirebaseCrash reporting initialized  
com.google.firebaseio.crash.internal.zzz@3333d325 07-20 08:57:24.442 D/FirebaseApp: Initialized class  
com.google.firebaseio.crash.FirebaseCrash.
```

And then when it sent the exception:

```
07-20 08:57:47.052 D/FirebaseCrashApiImpl: throwable java.lang.Exception: My first Android non-  
fatal error 07-20 08:58:18.822 D/FirebaseCrashSenderServiceImpl: Response code: 200 07-20  
08:58:18.822 D/FirebaseCrashSenderServiceImpl: Report sent
```

You can add custom logs to your report with

```
FirebaseCrash.log("Activity created");
```

Section 233.2: How to add Firebase Crash Reporting to your app

In order to add Firebase Crash Reporting to your app, perform the following steps:

- Create an app on the *Firebase Console* [here](#).
- Copy the `google-services.json` file from your project into your `app/` directory.
- Add the following rules to your root-level `build.gradle` file in order to include the `google-services` plugin:

```
buildscript {  
    // ...  
    dependencies {  
        // ...  
        classpath 'com.google.gms:google-services:3.0.0'  
    }  
}
```

- In your module Gradle file, add the `apply plugin` line at the bottom of the file to enable the Gradle plugin:

```
apply plugin: 'com.google.gms.google-services'
```

- 在应用级别的build.gradle文件中添加Crash Reporting的依赖：

```
compile 'com.google.firebaseio:firebase-crash:10.2.1'
```

- 然后你可以通过以下代码行从应用中触发自定义异常：

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

所有致命异常都会被报告到你的Firebase Console。

- 如果您想向控制台添加自定义日志，可以使用以下代码：

```
FirebaseCrash.log("第2级完成。");
```

欲了解更多信息，请访问：

- [官方文档](#)
- Stack Overflow 专题

- Add the dependency for Crash Reporting to your app-level build.gradle file:

```
compile 'com.google.firebaseio:firebase-crash:10.2.1'
```

- You can then fire a custom exception from your application by using the following line:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

All your fatal exceptions will be reported to your Firebase Console.

- If you want to add custom logs to a console, you can use the following code:

```
FirebaseCrash.log("Level 2 completed.");
```

For more information, please visit:

- [Official documentation](#)
- Stack Overflow dedicated topic

第234章：Twitter API

第234.1节：创建带有回调的Twitter登录按钮

1. 在您的布局中，添加一个登录按钮，代码如下：

```
<com.twitter.sdk.android.core.identity.TwitterLoginButton  
    android:id="@+id/twitter_login_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"/>
```

2. 在显示按钮的 Activity 或 Fragment 中，您需要创建并附加一个回调到登录按钮，具体如下：

```
import com.twitter.sdk.android.core.Callback;  
import com.twitter.sdk.android.core.Result;  
import com.twitter.sdk.android.core.TwitterException;  
import com.twitter.sdk.android.core.TwitterSession;  
import com.twitter.sdk.android.core.identity.TwitterLoginButton;  
  
...  
  
loginButton = (TwitterLoginButton) findViewById(R.id.login_button);  
loginButton.setCallback(new Callback<TwitterSession>() {  
    @Override  
    public void success(Result<TwitterSession> result) {  
        Log.d(TAG, "userName: " + session.getUserName());  
        Log.d(TAG, "userId: " + session.getUserId());  
        Log.d(TAG, "authToken: " + session.getAuthToken());  
        Log.d(TAG, "id: " + session.getId());  
        Log.d(TAG, "authToken: " + session.getAuthToken().token);  
        Log.d(TAG, "authSecret: " + session.getAuthToken().secret);  
    }  
  
    @Override  
    public void failure(TwitterException exception) {  
        // 失败时执行某些操作  
    }  
});
```

3. 将认证活动的结果传回按钮：

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    // 确保 loginButton 能接收到它触发的任何  
    // 活动的结果。  
    loginButton.onActivityResult(requestCode, resultCode, data);  
}
```

注意，如果在 Fragment 中使用 TwitterLoginButton，请改用以下步骤：

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

Chapter 234: Twitter APIs

Section 234.1: Creating login with twitter button and attach a callback to it

1. Inside your layout, add a Login button with the following code:

```
<com.twitter.sdk.android.core.identity.TwitterLoginButton  
    android:id="@+id/twitter_login_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"/>
```

2. In the Activity or Fragment that displays the button, you need to create and attach a Callback to the Login Button as the following:

```
import com.twitter.sdk.android.core.Callback;  
import com.twitter.sdk.android.core.Result;  
import com.twitter.sdk.android.core.TwitterException;  
import com.twitter.sdk.android.core.TwitterSession;  
import com.twitter.sdk.android.core.identity.TwitterLoginButton;  
  
...  
  
loginButton = (TwitterLoginButton) findViewById(R.id.login_button);  
loginButton.setCallback(new Callback<TwitterSession>() {  
    @Override  
    public void success(Result<TwitterSession> result) {  
        Log.d(TAG, "userName: " + session.getUserName());  
        Log.d(TAG, "userId: " + session.getUserId());  
        Log.d(TAG, "authToken: " + session.getAuthToken());  
        Log.d(TAG, "id: " + session.getId());  
        Log.d(TAG, "authToken: " + session.getAuthToken().token);  
        Log.d(TAG, "authSecret: " + session.getAuthToken().secret);  
    }  
  
    @Override  
    public void failure(TwitterException exception) {  
        // Do something on failure  
    }  
});
```

3. Pass the result of the authentication Activity back to the button:

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    // Make sure that the loginButton hears the result from any  
    // Activity that it triggered.  
    loginButton.onActivityResult(requestCode, resultCode, data);  
}
```

Note, If using the TwitterLoginButton in a Fragment, use the following steps instead:

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
super.onActivityResult(requestCode, resultCode, data);

// 将活动结果传递给片段，片段随后会将结果传递给登录按钮。

Fragment fragment = getFragmentManager().findFragmentById(R.id.your_fragment_id);
if (fragment != null) {
    fragment.onActivityResult(requestCode, resultCode, data);
}
}
```

4. 将以下行添加到你的 build.gradle 依赖项中：

```
apply plugin: 'io.fabric'

repositories {
    maven { url 'https://maven.fabric.io/public' }
}

compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
    transitive = true;
}
```

```
super.onActivityResult(requestCode, resultCode, data);

// Pass the activity result to the fragment, which will then pass the result to the login
// button.

Fragment fragment = getFragmentManager().findFragmentById(R.id.your_fragment_id);
if (fragment != null) {
    fragment.onActivityResult(requestCode, resultCode, data);
}
}
```

4. Add the following lines to your **build.gradle** dependencies:

```
apply plugin: 'io.fabric'

repositories {
    maven { url 'https://maven.fabric.io/public' }
}

compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
    transitive = true;
}
```

第235章：Youtube-API

第235.1节：扩展YouTubeBaseActivity的活动

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
        mPlayerView.initialize(mApiKey, this); // 设置OnInitializedListener
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
            LayoutParams.MATCH_PARENT)); // 全屏显示
    }

    // 当播放器初始化成功时调用。
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
        YouTubePlayer player,
        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // 设置播放器状态变化监听器
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
        YouTubeInitializationResult errorReason) {
        Toast.makeText(this, "初始化时出错", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { // 视频已加载
        if (!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // 如果不调用play，视频不会自动播放，但用户仍可通过播放按钮播放
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }
}
```

Chapter 235: Youtube-API

Section 235.1: Activity extending YouTubeBaseActivity

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
        mPlayerView.initialize(mApiKey, this); // setting up OnInitializedListener
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
            LayoutParams.MATCH_PARENT)); // show it in full screen
    }

    // Called when initialization of the player succeeds.
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
        YouTubePlayer player,
        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // setting up the player state change listener
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
        YouTubeInitializationResult errorReason) {
        Toast.makeText(this, "Error While initializing", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { // video has been loaded
        if (!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // if we don't call play then video will not auto play, but user
        still has the option to play via play button
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }
}
```

```

@Override
public void onVideoStarted() {
}

@Override
public void onError(ErrorReason reason) {
    Log.e("onError", "错误：" + reason.name());
}
}

```

```

@Override
public void onVideoStarted() {
}

@Override
public void onError(ErrorReason reason) {
    Log.e("onError", "onError：" + reason.name());
}
}

```

第235.2节：在Android上使用YouTube数据API

本示例将指导您如何在安卓上使用YouTube数据API获取播放列表数据。

SHA-1 指纹

首先，您需要获取机器的 SHA-1 指纹。获取方法有多种，您可以选择本问答中提供的任何一种方法。

Google API 控制台和 Android 的 YouTube 密钥

现在您已经有了 SHA-1 指纹，打开 Google API 控制台并创建一个项目。访问此页面，使用该 [SHA-1 密钥](#) 创建项目并启用 YouTube 数据 API。随后您将获得一个密钥，该密钥将用于从 Android 发送请求并获取数据。

Gradle 部分

您需要在 Gradle 文件中添加以下行以使用 YouTube 数据 API：

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

为了使用 YouTube 的原生客户端发送请求，我们必须在 Gradle 中添加以下行：

```
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
```

为了避免冲突，还需要在 Gradle 中添加以下配置：

```
配置。全部 {
    解析策略。强制 'com.google.code.findbugs:jsr305:3.0.2'
}
```

下面展示了最终的gradle.build文件的样子。

build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        应用ID "com.aam.skillschool"
        最低SDK版本 19
    目标SDK版本 25
}
}

```

Section 235.2: Consuming YouTube Data API on Android

This example will guide you how to get playlist data using the YouTube Data API on Android.

SHA-1 fingerprint

First you need to get an SHA-1 fingerprint for your machine. There are various methods for retrieving it. You can choose any method provided in [this Q&A](#).

Google API console and YouTube key for Android

Now that you have an SHA-1 fingerprint, open the Google API console and create a project. Go to [this page](#) and create a project using that SHA-1 key and enable the YouTube Data API. Now you will get a key. This key will be used to send requests from Android and fetch data.

Gradle part

You will have to add the following lines to your Gradle file for the YouTube Data API:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

In order to use YouTube's native client to send requests, we have to add the following lines in Gradle:

```
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
```

The following configuration also needs to be added in Gradle in order to avoid conflicts:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Below it is shown how the `gradle.build` would finally look like.

build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        应用ID "com.aam.skillschool"
        minSdkVersion 19
        targetSdkVersion 25
}
}

```

```

版本号 1
versionName "1.0"
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
配置。全部 {
解析策略。强制 'com.google.code.findbugs:jsr305:3.0.2'
}
}

dependencies {
编译文件树(包含: ['*.jar'], 目录: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        排除组: 'com.android.support', 模块: 'support-annotations'
    })
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.http-client:google-http-client-android:+'
    compile 'com.google.api-client:google-api-client-android:+'
    compile 'com.google.api-client:google-api-client-gson:+'
}

```

现在进入Java部分。由于我们将使用HttpTransport进行网络通信，使用GsonFactory将JSON转换为POJO，因此不需要其他库来发送请求。

现在我想展示如何通过提供播放列表ID来使用YouTube API获取播放列表。为此任务我将使用 AsyncTask。要了解如何请求参数以及理解流程，请查看 [YouTube Data API](#)。

```

public class GetPlaylistDataAsyncTask extends AsyncTask<String[], Void, PlaylistListResponse> {
    private static final String YOUTUBE_PLAYLIST_PART = "snippet";
    private static final String YOUTUBE_PLAYLIST_FIELDS = "items(id,snippet(title))";

    private YouTube mYouTubeDataApi;

    public GetPlaylistDataAsyncTask(YouTube api) {
        mYouTubeDataApi = api;
    }

    @Override
    protected PlaylistListResponse doInBackground(String[]... params) {

        final String[] playlistIds = params[0];

        PlaylistListResponse playlistListResponse;
        try {
            playlistListResponse = mYouTubeDataApi.playlists()
                .list(YOUTUBE_PLAYLIST_PART)
                .setId(TextUtils.join(", ", playlistIds))
                .setFields(YOUTUBE_PLAYLIST_FIELDS)
                .setKey(AppConstants.YOUTUBE_KEY) //这里你需要提供密钥
                .execute();
        } 捕获 (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
}

```

```

versionCode 1
versionName "1.0"
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
}

dependencies {
compile fileTree(include: ['*.jar'], dir: 'libs')
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
    exclude group: 'com.android.support', module: 'support-annotations'
})
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
compile 'com.android.support:appcompat-v7:25.3.1'
compile 'com.android.support:support-v4:25.3.1'
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
}

```

Now comes the Java part. Since we will be using HttpTransport for networking and GsonFactory for converting JSON into POJO, we don't need any other library to send any requests.

Now I want to show how to get playlists via the YouTube API by providing the playlist IDs. For this task I will use AsyncTask. To understand how we request parameters and to understand the flow, please take a look at the [YouTube Data API](#).

```

public class GetPlaylistDataAsyncTask extends AsyncTask<String[], Void, PlaylistListResponse> {
    private static final String YOUTUBE_PLAYLIST_PART = "snippet";
    private static final String YOUTUBE_PLAYLIST_FIELDS = "items(id,snippet(title))";

    private YouTube mYouTubeDataApi;

    public GetPlaylistDataAsyncTask(YouTube api) {
        mYouTubeDataApi = api;
    }

    @Override
    protected PlaylistListResponse doInBackground(String[]... params) {

        final String[] playlistIds = params[0];

        PlaylistListResponse playlistListResponse;
        try {
            playlistListResponse = mYouTubeDataApi.playlists()
                .list(YOUTUBE_PLAYLIST_PART)
                .setId(TextUtils.join(", ", playlistIds))
                .setFields(YOUTUBE_PLAYLIST_FIELDS)
                .setKey(AppConstants.YOUTUBE_KEY) //Here you will have to provide the keys
                .execute();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
}

```

```

    }

    return playlistListResponse;
}

}

```

上述异步任务将返回一个PlaylistListResponse实例，这是YouTube SDK内置的类。它包含所有必需的字段，因此我们不必自己创建POJO。

最后，在我们的MainActivity中需要执行以下操作：

```

public class MainActivity extends AppCompatActivity {
    private YouTube mYoutubeDataApi;
    private final GsonFactory mJsonFactory = new GsonFactory();
    private final HttpTransport mTransport = AndroidHttp.newCompatibleTransport();
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_review);
        mYoutubeDataApi = new YouTube.Builder(mTransport, mJsonFactory, null)
            .setApplicationName(getResources().getString(R.string.app_name))
            .build();
        String[] ids = {"some playlists ids here separated by ,,"};
        new GetPlaylistDataTask(mYoutubeDataApi) {
            ProgressDialog progressDialog = new ProgressDialog(getActivity());
            ...
        }.execute(ids);
    }
}

```

第235.3节：启动独立播放器活动

1. 启动独立播放器活动

```

Intent standAlonePlayerIntent = YouTubeStandalonePlayer.createVideoIntent((Activity)
    context,
    Config.YOUTUBE_API_KEY, // 你在步骤3中创建的
    videoId, // 要播放的视频
    100, // 视频播放开始的时间，单位为毫秒
    true, // 是否自动播放
    false); // 是否为灯箱模式；false表示在全屏上下文中显示
context.startActivity(standAlonePlayerIntent);

```

```

    }

    return playlistListResponse;
}

}

```

The above asynchronous task will return an instance of PlaylistListResponse which is a build-in class of the YouTube SDK. It has all the required fields, so we don't have to create POJOs ourself.

Finally, in our MainActivity we will have to do the following:

```

public class MainActivity extends AppCompatActivity {
    private YouTube mYoutubeDataApi;
    private final GsonFactory mJsonFactory = new GsonFactory();
    private final HttpTransport mTransport = AndroidHttp.newCompatibleTransport();
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_review);
        mYoutubeDataApi = new YouTube.Builder(mTransport, mJsonFactory, null)
            .setApplicationName(getResources().getString(R.string.app_name))
            .build();
        String[] ids = {"some playlists ids here separated by ,,"};
        new GetPlaylistDataTask(mYoutubeDataApi) {
            ProgressDialog progressDialog = new ProgressDialog(getActivity());
            ...
        }.execute(ids);
    }
}

```

Section 235.3: Launching StandalonePlayerActivity

1. Launch standalone player activity

```

Intent standAlonePlayerIntent = YouTubeStandalonePlayer.createVideoIntent((Activity)
    context,
    Config.YOUTUBE_API_KEY, // which you have created in step 3
    videoId, // video which is to be played
    100, // The time, in milliseconds, where playback should start in the video
    true, // autoplay or not
    false); //lightbox mode or not; false will show in fullscreen
context.startActivity(standAlonePlayerIntent);

```

第235.4节：竖屏活动中的YoutubePlayerFragment

以下代码实现了一个简单的YoutubePlayerFragment。活动的布局被锁定为竖屏模式
当方向改变或用户点击YoutubePlayer的全屏时，界面切换为横屏，
YoutubePlayer填满屏幕。YoutubePlayerFragment不需要继承Youtube库提供的活动。
它需要实现YouTubePlayer.OnInitializedListener接口以初始化YoutubePlayer。
因此，我们的活动类如下所示

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import com.google.android.youtube.player.YouTubeInitializationResult;
import com.google.android.youtube.player.YouTubePlayer;
import com.google.android.youtube.player.YouTubePlayerFragment;

public class MainActivity extends AppCompatActivity implements YouTubePlayer.OnInitializedListener {

    public static final String API_KEY ;
    public static final String VIDEO_ID = "B08iLAtS3AQ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment) getSupportFragmentManager()
            .findFragmentById(R.id.youtubeplayerfragment);

        youTubePlayerFragment.initialize(API_KEY, this);

    }

    /**
     *
     * @param provider 用于初始化 YouTubePlayer 的提供者      * @param youTubePlayer 一个可用于控制视频播放的 YouTubePlayer 对象，属于该提供者。
     *
     * @param wasRestored 指示播放器是否从之前保存的状态恢复，作为 YouTubePlayerView
     *
     * 或 YouTubePlayerFragment 恢复其状态的一部分。true 通常表示播放正在从用户预期的位置继续
     * ，且不应加载新视频
     */
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, boolean wasRestored) {

        youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
            YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);

        if(!wasRestored) {
            youTubePlayer.cueVideo(VIDEO_ID);
        }
    }

    /**
     *
     * @param provider 用于初始化 YouTubePlayer 的提供者      * @param youTubePlayer 一个可用于控制视频播放的 YouTubePlayer 对象，属于该提供者。
     *
     * @param wasRestored 指示播放器是否从之前保存的状态恢复，作为 YouTubePlayerView
     *
     * 或 YouTubePlayerFragment 恢复其状态的一部分。true 通常表示播放正在从用户预期的位置继续
     * ，且不应加载新视频
     */
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, YouTubeInitializationResult error) {
    }
}
```

Section 235.4: YoutubePlayerFragment in portrait Activity

The following code implements a simple YoutubePlayerFragment. The activity's layout is locked in portrait mode and when orientation changes or the user clicks full screen at the YoutubePlayer it turns to landscape with the YoutubePlayer filling the screen. The YoutubePlayerFragment does not need to extend an activity provided by the Youtube library. It needs to implement YouTubePlayer.OnInitializedListener in order to get the YoutubePlayer initialized. So our Activity's class is the following

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import com.google.android.youtube.player.YouTubeInitializationResult;
import com.google.android.youtube.player.YouTubePlayer;
import com.google.android.youtube.player.YouTubePlayerFragment;

public class MainActivity extends AppCompatActivity implements YouTubePlayer.OnInitializedListener {

    public static final String API_KEY ;
    public static final String VIDEO_ID = "B08iLAtS3AQ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment) getSupportFragmentManager()
            .findFragmentById(R.id.youtubeplayerfragment);

        youTubePlayerFragment.initialize(API_KEY, this);

    }

    /**
     *
     * @param provider 用于初始化 YouTubePlayer 的提供者      * @param youTubePlayer 一个可用于控制视频播放的 YouTubePlayer 对象，属于该提供者。
     *
     * @param wasRestored 指示播放器是否从之前保存的状态恢复，作为 YouTubePlayerView
     *
     * 或 YouTubePlayerFragment 恢复其状态的一部分。true 通常表示播放正在从用户预期的位置继续
     * ，且不应加载新视频
     */
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, boolean wasRestored) {

        youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
            YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);

        if(!wasRestored) {
            youTubePlayer.cueVideo(VIDEO_ID);
        }
    }

    /**
     *
     * @param provider 用于初始化 YouTubePlayer 的提供者      * @param youTubePlayer 一个可用于控制视频播放的 YouTubePlayer 对象，属于该提供者。
     *
     * @param wasRestored 指示播放器是否从之前保存的状态恢复，作为 YouTubePlayerView
     *
     * 或 YouTubePlayerFragment 恢复其状态的一部分。true 通常表示播放正在从用户预期的位置继续
     * ，且不应加载新视频
     */
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, YouTubeInitializationResult error) {
    }
}
```

```

*
* @param provider 初始化 YouTubePlayer 失败的提供者。
* @param error 失败的原因，以及该失败的潜在解决方案。
*/
@Override
public void onInitializationFailure(YouTubePlayer.Provider provider,
YouTubeInitializationResult error) {

    final int REQUEST_CODE = 1;

    if(error.isUserRecoverableError()) {
        error.getErrorDialog(this,REQUEST_CODE).show();
    } else {
        String errorMessage = String.format("初始化 YoutubePlayer 时出错 (%1$s)", error.toString());
        Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
    }
}

```

可以将 YoutubePlayerFragment 添加到活动的布局 xaml 中，方法如下

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/youtubeplayerfragment"
        android:name="com.google.android.youtube.player.YouTubePlayerFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            &lt;TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="这是一个 YoutubePlayerFragment 示例"
                android:textStyle="bold"/>

            &lt;TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

```

```

*
* @param provider The provider which failed to initialize a YouTubePlayer.
* @param error The reason for this failure, along with potential resolutions to this failure.
*/
@Override
public void onInitializationFailure(YouTubePlayer.Provider provider,
YouTubeInitializationResult error) {

    final int REQUEST_CODE = 1;

    if(error.isUserRecoverableError()) {
        error.getErrorDialog(this,REQUEST_CODE).show();
    } else {
        String errorMessage = String.format("There was an error initializing the YoutubePlayer (%1$s)", error.toString());
        Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
    }
}

```

A YoutubePlayerFragment can be added to the activity's layout xaml as followed

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/youtubeplayerfragment"
        android:name="com.google.android.youtube.player.YouTubePlayerFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

```

```
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="这是一个 YoutubePlayerFragment 示例"
    android:textStyle="bold"/>
```

```
    </LinearLayout>
</ScrollView>
```

```
</LinearLayout>
```

最后，您需要在清单文件中活动标签内添加以下属性

```
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:screenOrientation="portrait"
```

第235.5节：YouTube播放器API

获取Android API密钥：

首先，您需要使用java keytool在您的机器上获取SHA-1指纹。请在cmd/终端中执行以下命令以获取SHA-1指纹。

```
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
&lt;TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:text="This is a YoutubePlayerFragment example"
    android:textStyle="bold"/>
```

```
    </LinearLayout>
</ScrollView>
```

```
</LinearLayout>
```

Lastly you need to add the following attributes in your Manifest file inside the activity's tag

```
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:screenOrientation="portrait"
```

Section 235.5: YouTube Player API

Obtaining the Android API Key :

First you'll need to get the SHA-1 fingerprint on your machine using java keytool. Execute the below command in cmd/terminal to get the SHA-1 fingerprint.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

MainActivity.java

```
public class Activity extends YouTubeBaseActivity implements YouTubePlayer.OnInitializedListener {  
  
    private static final int RECOVERY_DIALOG_REQUEST = 1;  
  
    // YouTube播放器视图  
    private YouTubePlayerView youTubeView;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,  
                            WindowManager.LayoutParams.FLAG_FULLSCREEN);  
  
        setContentView(R.layout.activity_main);  
  
        youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_view);  
  
        // 使用开发者密钥初始化视频播放器  
        youTubeView.initialize(Config.DEVELOPER_KEY, this);  
    }  
  
    @Override  
    public void onInitializationFailure(YouTubePlayer.Provider provider,  
                                       YouTubeInitializationResult errorReason) {  
        if (errorReason.isUserRecoverableError()) {  
            errorReason.getErrorDialog(this, RECOVERY_DIALOG_REQUEST).show();  
        } else {  
            String errorMessage = String.format(  
getString(R.string.error_player), errorReason.toString());  
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();  
        }  
    }  
  
    @Override  
    public void onInitializationSuccess(YouTubePlayer.Provider provider,  
                                       YouTubePlayer player, boolean wasRestored) {  
        if (!wasRestored) {  
  
            // loadVideo() 将自动播放视频  
            // 如果不想自动播放, 请使用 cueVideo() 方法  
            player.loadVideo(Config.YOUTUBE_VIDEO_CODE);  
  
            // 隐藏播放器控件  
            player.setPlayerStyle(YouTubePlayer.PlayerStyle.CHROMELESS);  
        }  
    }  
  
    @Override  
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
        if (requestCode == RECOVERY_DIALOG_REQUEST) {  
            // 如果用户执行了恢复操作, 则重试初始化  
            getYouTubePlayerProvider().initialize(Config.DEVELOPER_KEY, this);  
        }  
    }  
}
```

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

MainActivity.java

```
public class Activity extends YouTubeBaseActivity implements YouTubePlayer.OnInitializedListener {  
  
    private static final int RECOVERY_DIALOG_REQUEST = 1;  
  
    // YouTube player view  
    private YouTubePlayerView youTubeView;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,  
                            WindowManager.LayoutParams.FLAG_FULLSCREEN);  
  
        setContentView(R.layout.activity_main);  
  
        youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_view);  
  
        // Initializing video player with developer key  
        youTubeView.initialize(Config.DEVELOPER_KEY, this);  
    }  
  
    @Override  
    public void onInitializationFailure(YouTubePlayer.Provider provider,  
                                       YouTubeInitializationResult errorReason) {  
        if (errorReason.isUserRecoverableError()) {  
            errorReason.getErrorDialog(this, RECOVERY_DIALOG_REQUEST).show();  
        } else {  
            String errorMessage = String.format(  
getString(R.string.error_player), errorReason.toString());  
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();  
        }  
    }  
  
    @Override  
    public void onInitializationSuccess(YouTubePlayer.Provider provider,  
                                       YouTubePlayer player, boolean wasRestored) {  
        if (!wasRestored) {  
  
            // loadVideo() will auto play video  
            // Use cueVideo() method, if you don't want to play it automatically  
            player.loadVideo(Config.YOUTUBE_VIDEO_CODE);  
  
            // Hiding player controls  
            player.setPlayerStyle(YouTubePlayer.PlayerStyle.CHROMELESS);  
        }  
    }  
  
    @Override  
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
        if (requestCode == RECOVERY_DIALOG_REQUEST) {  
            // Retry initialization if user performed a recovery action  
            getYouTubePlayerProvider().initialize(Config.DEVELOPER_KEY, this);  
        }  
    }  
}
```

```
private YouTubePlayer.Provider getYoutubePlayerProvider() {  
    return (YouTubePlayerView) findViewById(R.id.youtube_view);  
}  
}
```

现在创建 Config.java 文件。该文件保存 Google 控制台 API 开发者密钥和 YouTube 视频 ID

Config.java

```
public class Config {  
  
    // 开发者密钥  
    public static final String DEVELOPER_KEY = "AIzaSyDZtE10od_hXM5aXYEh6Zn7c6brV9ZjKuk";  
  
    // YouTube 视频 ID  
    public static final String YOUTUBE_VIDEO_CODE = "_oEA18Y8gM0";  
}
```

xml 文件

```
<com.google.android.youtube.player.YouTubePlayerView  
    android:id="@+id/youtube_view"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="30dp" />
```

```
private YouTubePlayer.Provider getYoutubePlayerProvider() {  
    return (YouTubePlayerView) findViewById(R.id.youtube_view);  
}
```

Now create Config.java file. This file holds the Google Console API developer key and YouTube video id

Config.java

```
public class Config {  
  
    // Developer key  
    public static final String DEVELOPER_KEY = "AIzaSyDZtE10od_hXM5aXYEh6Zn7c6brV9ZjKuk";  
  
    // YouTube video id  
    public static final String YOUTUBE_VIDEO_CODE = "_oEA18Y8gM0";  
}
```

xml file

```
<com.google.android.youtube.player.YouTubePlayerView  
    android:id="@+id/youtube_view"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="30dp" />
```

第236章：集成谷歌登录

参数	详情
标签	用于日志记录的字符串
GoogleSignInHelper	一个静态的辅助引用
AppCompatActivity	一个活动引用
GoogleApiClient	GoogleAPIClient 的一个引用
RC_SIGN_IN	一个整数，表示活动结果常量
isLoggingOut	一个布尔值，用于检查注销任务是否正在运行

第236.1节：使用辅助类的谷歌登录

在你的build.gradle中，android标签外添加以下内容：

```
// 将插件应用到应用程序。  
apply plugin: 'com.google.gms.google-services'
```

将以下辅助类添加到你的util包中：

```
/**  
 * 由Andy创建  
 */  
  
public class GoogleSignInHelper implements GoogleApiClient.OnConnectionFailedListener,  
    GoogleApiClient.ConnectionCallbacks {  
    private static final String TAG = GoogleSignInHelper.class.getSimpleName();  
  
    private static GoogleSignInHelper googleSignInHelper;  
    private AppCompatActivity mActivity;  
    private GoogleApiClient mGoogleApiClient;  
    public static final int RC_SIGN_IN = 9001;  
    private boolean isLoggingOut = false;  
  
    public static GoogleSignInHelper newInstance(AppCompatActivity mActivity) {  
        if (googleSignInHelper == null) {  
            googleSignInHelper = new GoogleSignInHelper(mActivity, fireBaseAuthHelper);  
        }  
        return googleSignInHelper;  
    }  
  
    public GoogleSignInHelper(AppCompatActivity mActivity) {  
        this.mActivity = mActivity;  
        initGoogleSignIn();  
    }  
  
    private void initGoogleSignIn() {  
        // [START config_sign_in]  
        // Configure Google Sign In  
        GoogleSignInOptions gso = new  
        GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)  
            .requestIdToken(mActivity.getString(R.string.default_web_client_id))  
            .requestEmail()  
.build();  
        // [END config_sign_in]  
  
        mGoogleApiClient = new GoogleApiClient.Builder(mActivity)  
            .enableAutoManage(mActivity /* FragmentActivity */, this /*  
            OnConnectionFailedListener */)  
            .addConnectionCallbacks(this /* ConnectionCallbacks */)  
            .addOnConnectionFailedListener(this /* OnConnectionFailedListener */);  
    }  
  
    public void signOut() {  
        if (mGoogleApiClient != null) {  
            mGoogleApiClient.disconnect();  
        }  
    }  
  
    public void signIn() {  
        if (mGoogleApiClient != null) {  
            Intent signInIntent = GoogleSignIn.getClient(mActivity, gso).getSignInIntent();  
            startActivityForResult(signInIntent, RC_SIGN_IN);  
        }  
    }  
  
    @Override  
    public void onConnectionFailed(ConnectionResult connectionResult) {  
        Log.d(TAG, "onConnectionFailed: " + connectionResult);  
    }  
  
    @Override  
    public void onConnected(Bundle bundle) {  
        Log.d(TAG, "onConnected");  
    }  
  
    @Override  
    public void onConnectionSuspended(int i) {  
        Log.d(TAG, "onConnectionSuspended: " + i);  
    }  
}
```

Chapter 236: Integrate Google Sign In

Parameter	Detail
TAG	A String used while logging
GoogleSignInHelper	A static reference for helper
AppCompatActivity	An Activity reference
GoogleApiClient	A reference of GoogleAPIClient
RC_SIGN_IN	An integer represents activity result constant
isLoggingOut	A boolean to check if log-out task is running or not

Section 236.1: Google Sign In with Helper class

Add below to your build.gradle out of android tag:

```
// Apply plug-in to app.  
apply plugin: 'com.google.gms.google-services'
```

Add below helper class to your util package:

```
/**  
 * Created by Andy  
 */  
  
public class GoogleSignInHelper implements GoogleApiClient.OnConnectionFailedListener,  
    GoogleApiClient.ConnectionCallbacks {  
    private static final String TAG = GoogleSignInHelper.class.getSimpleName();  
  
    private static GoogleSignInHelper googleSignInHelper;  
    private AppCompatActivity mActivity;  
    private GoogleApiClient mGoogleApiClient;  
    public static final int RC_SIGN_IN = 9001;  
    private boolean isLoggingOut = false;  
  
    public static GoogleSignInHelper newInstance(AppCompatActivity mActivity) {  
        if (googleSignInHelper == null) {  
            googleSignInHelper = new GoogleSignInHelper(mActivity, fireBaseAuthHelper);  
        }  
        return googleSignInHelper;  
    }  
  
    public GoogleSignInHelper(AppCompatActivity mActivity) {  
        this.mActivity = mActivity;  
        initGoogleSignIn();  
    }  
  
    private void initGoogleSignIn() {  
        // [START config_sign_in]  
        // Configure Google Sign In  
        GoogleSignInOptions gso = new  
        GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)  
            .requestIdToken(mActivity.getString(R.string.default_web_client_id))  
            .requestEmail()  
.build();  
        // [END config_sign_in]  
  
        mGoogleApiClient = new GoogleApiClient.Builder(mActivity)  
            .enableAutoManage(mActivity /* FragmentActivity */, this /*  
            OnConnectionFailedListener */)  
            .addConnectionCallbacks(this /* ConnectionCallbacks */)  
            .addOnConnectionFailedListener(this /* OnConnectionFailedListener */);  
    }  
  
    public void signOut() {  
        if (mGoogleApiClient != null) {  
            mGoogleApiClient.disconnect();  
        }  
    }  
  
    public void signIn() {  
        if (mGoogleApiClient != null) {  
            Intent signInIntent = GoogleSignIn.getClient(mActivity, gso).getSignInIntent();  
            startActivityForResult(signInIntent, RC_SIGN_IN);  
        }  
    }  
  
    @Override  
    public void onConnectionFailed(ConnectionResult connectionResult) {  
        Log.d(TAG, "onConnectionFailed: " + connectionResult);  
    }  
  
    @Override  
    public void onConnected(Bundle bundle) {  
        Log.d(TAG, "onConnected");  
    }  
  
    @Override  
    public void onConnectionSuspended(int i) {  
        Log.d(TAG, "onConnectionSuspended: " + i);  
    }  
}
```

```

OnConnectionFailedListener */
.addApi(Auth.GOOGLE_SIGN_IN_API, gso)
    .addConnectionCallbacks(this)
.build();

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    // 发生了无法解决的错误, Google API (包括登录) 将无法使用。
    Log.d(TAG, "onConnectionFailed:" + connectionResult);
    Toast.makeText(mActivity, "Google Play 服务错误。", Toast.LENGTH_SHORT).show();
}

public void 获取Google账户详情(GoogleSignInResult result) {
    // Google 登录成功, 使用 Firebase 进行认证
    GoogleSignInAccount account = result.getSignInAccount();
    // 您现在已登录 Google
}
public void signOut() {

    if (mGoogleApiClient.isConnected()) {

        // Google 登出
        Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
            new ResultCallback<Status>() {
                @Override
                public void onResult(@NonNull Status status) {
                    isLoggingOut = false;
                }
            });
    } else {
        isLoggingOut = true;
    }
}

public GoogleApiClient getGoogleClient() {
    return mGoogleApiClient;
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    Log.w(TAG, "onConnected");
    if (isLoggingOut) {
        signOut();
    }
}

@Override
public void onConnectionSuspended(int i) {
    Log.w(TAG, "onConnectionSuspended");
}
}

```

在 Activity 文件中的 OnActivityResult 方法中添加以下代码：

```

// [START onactivityresult]
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
}

```

```

OnConnectionFailedListener */
.addApi(Auth.GOOGLE_SIGN_IN_API, gso)
    .addConnectionCallbacks(this)
.build();

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    // An unresolvable error has occurred and Google APIs (including Sign-In) will not
    // be available.
    Log.d(TAG, "onConnectionFailed:" + connectionResult);
    Toast.makeText(mActivity, "Google Play Services error.", Toast.LENGTH_SHORT).show();
}

public void getGoogleAccountDetails(GoogleSignInResult result) {
    // Google Sign In was successful, authenticate with Firebase
    GoogleSignInAccount account = result.getSignInAccount();
    // You are now logged into Google
}
public void signOut() {

    if (mGoogleApiClient.isConnected()) {

        // Google sign out
        Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
            new ResultCallback<Status>() {
                @Override
                public void onResult(@NonNull Status status) {
                    isLoggingOut = false;
                }
            });
    } else {
        isLoggingOut = true;
    }
}

public GoogleApiClient getGoogleClient() {
    return mGoogleApiClient;
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    Log.w(TAG, "onConnected");
    if (isLoggingOut) {
        signOut();
    }
}

@Override
public void onConnectionSuspended(int i) {
    Log.w(TAG, "onConnectionSuspended");
}
}

```

Add below code to your OnActivityResult in Activity file:

```

// [START onactivityresult]
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
}

```

```

// 从 GoogleSignInApi.getSignInIntent(...) 启动的 Intent 返回的结果
if (requestCode == GoogleSignInHelper.RC_SIGN_IN) {
    GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
    if (result.isSuccess()) {
        googleSignInHelper.getGoogleAccountDetails(result);
    } else {
        // Google 登录失败, 适当更新 UI
        // [START_EXCLUDE]
        Log.d(TAG, "使用 Google 登录失败");
        // [END_EXCLUDE]
    }
}
// [END onactivityresult]

// [START signin]
public void signIn() {
Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(googleSignInHelper.getGoogleClient());
startActivityForResult(signInIntent, GoogleSignInHelper.RC_SIGN_IN);
}

// [END signin]

```

```

// Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
if (requestCode == GoogleSignInHelper.RC_SIGN_IN) {
    GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
    if (result.isSuccess()) {
        googleSignInHelper.getGoogleAccountDetails(result);
    } else {
        // Google Sign In failed, update UI appropriately
        // [START_EXCLUDE]
        Log.d(TAG, "signInWith Google failed");
        // [END_EXCLUDE]
    }
}
// [END onactivityresult]

// [START signin]
public void signIn() {
Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(googleSignInHelper.getGoogleClient());
startActivityForResult(signInIntent, GoogleSignInHelper.RC_SIGN_IN);
}

// [END signin]

```

第237章：Android上的Google登录集成

本主题基于如何在Android应用中集成Google登录

第237.1节：在项目中集成谷歌身份验证。 (获取配置文件)

首先从以下链接获取登录的配置文件

[\[https://developers.google.com/identity/sign-in/android/start-integrating\]\[1\]](https://developers.google.com/identity/sign-in/android/start-integrating)

点击获取配置文件

- 输入应用名称和包名，点击选择并配置服务
- 提供SHA1启用谷歌登录并生成配置文件

下载配置文件并将文件放置在项目的app/文件夹中

1. 在项目级build.gradle中添加依赖：

```
classpath 'com.google.gms:google-services:3.0.0'
```

2. 在应用级build.gradle中添加插件： (底部)

```
apply plugin: 'com.google.gms.google-services'
```

3. 在应用的gradle文件中添加此依赖

```
dependencies { compile 'com.google.android.gms:play-services-auth:9.8.0' }
```

第237.2节：代码实现 Google 登录

- 在您的登录活动的 `onCreate` 方法中，配置 Google 登录以请求您的应用所需的用户数据。

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
.build();
```

- 创建一个具有访问 Google 登录 API 和您指定选项的 `GoogleApiClient` 对象。

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
    .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
.build();
```

- 现在，当用户点击 Google 登录按钮时，调用此函数。

Chapter 237: Google signin integration on android

This topic is based on How to integrate google sign-in, On android apps

Section 237.1: Integration of google Auth in your project. (Get a configuration file)

First get the Configuration File for Sign-in from

Open link below

[\[https://developers.google.com/identity/sign-in/android/start-integrating\]\[1\]](https://developers.google.com/identity/sign-in/android/start-integrating)

click on get A configuration file

- Enter App name And package name and click on choose and configure services
- provide SHA1 Enable google SIGNIN and generate configuration files

Download the configuration file and place the file in app/ folder of your project

1. Add the dependency to your project-level build.gradle:

```
classpath 'com.google.gms:google-services:3.0.0'
```

2. Add the plugin to your app-level build.gradle:(bottom)

```
apply plugin: 'com.google.gms.google-services'
```

3. add this dependency to your app gradle file

```
dependencies { compile 'com.google.android.gms:play-services-auth:9.8.0' }
```

Section 237.2: Code Implementation Google SignIn

- In your sign-in activity's `onCreate` method, configure Google Sign-In to request the user data required by your app.

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
.build();
```

- create a `GoogleApiClient` object with access to the Google Sign-In API and the options you specified.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
    .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
.build();
```

- Now When User click on Google signin button call this Function.

```
private void signIn() {
Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- 实现 OnActivityResult 以获取响应。

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);

// 从 GoogleSignInApi.getSignInIntent(...) 启动的 Intent 返回的结果
if (requestCode == RC_SIGN_IN) {
GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
handleSignInResult(result);
}
}
```

- 最后一步 处理结果并获取用户数据

```
private void handleSignInResult(GoogleSignInResult result) {
Log.d(TAG, "handleSignInResult:" + result.isSuccess());
if (result.isSuccess()) {
    // 登录成功, 显示已认证的用户界面。
    GoogleSignInAccount acct = result.getSignInAccount();
    mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
    updateUI(true);
} else {
    // 已登出, 显示未认证的用户界面。
    updateUI(false);
}
}
```

```
private void signIn() {
Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- implement OnActivityResult to get the response.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);

// Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
if (requestCode == RC_SIGN_IN) {
    GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
    handleSignInResult(result);
}
}
```

- Last step Handle The Result and get User Data

```
private void handleSignInResult(GoogleSignInResult result) {
Log.d(TAG, "handleSignInResult:" + result.isSuccess());
if (result.isSuccess()) {
    // Signed in successfully, show authenticated UI.
    GoogleSignInAccount acct = result.getSignInAccount();
    mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
    updateUI(true);
} else {
    // Signed out, show unauthenticated UI.
    updateUI(false);
}
}
```

第238章：谷歌感知API

第238.1节：使用围栏API获取特定范围内位置的变化

如果您想检测用户何时进入特定位置，可以为该位置创建一个带有指定半径的围栏，并在用户进入或离开该位置时收到通知。

```
// 您自己的动作过滤器，类似于Manifest中使用的那些
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "locationFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// 确保按照备注部分描述初始化您的客户端
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 等等

    // 0 是一个标准的Activity请求码，可以根据需要更改
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // 创建围栏
    AwarenessFence fence = LocationFence.entering(48.136334, 11.581660, 25);
    // 注册围栏以接收回调。
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
    .setResultCallback(new ResultCallback<Status>() {
        @Override
        public void onResult(@NonNull Status status) {
            if (status.isSuccess()) {
                Log.i(FENCE_KEY, "Successfully registered.");
            } else {
                Log.e(FENCE_KEY, "Could not be registered: " + status);
            }
        }
    });
}

现在创建一个 BroadcastReceiver 来接收用户状态的更新：
```

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // 获取围栏状态
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is in location");
                break;
        }
    }
}
```

Chapter 238: Google Awareness APIs

Section 238.1: Get changes for location within a certain range using Fence API

If you want to detect when your user enters a specific location, you can create a fence for the specific location with a radius you want and be notified when your user enters or leaves the location.

```
// Your own action filter, like the ones used in the Manifest
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "locationFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // etc

    // The 0 is a standard Activity request code that can be changed for your needs
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence
    AwarenessFence fence = LocationFence.entering(48.136334, 11.581660, 25);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
    .setResultCallback(new ResultCallback<Status>() {
        @Override
        public void onResult(@NonNull Status status) {
            if (status.isSuccess()) {
                Log.i(FENCE_KEY, "Successfully registered.");
            } else {
                Log.e(FENCE_KEY, "Could not be registered: " + status);
            }
        }
    });
}

Now create a BroadcastReciver to recive updates in user state:
```

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the fence state
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is in location");
                break;
        }
    }
}
```

```
        case FenceState.FALSE:  
            Log.i(TAG, "用户不在该位置");  
            break;  
        case FenceState.UNKNOWN:  
            Log.i(TAG, "用户正在执行未知操作");  
            break;  
    }  
}
```

第238.2节：使用Snapshot API获取当前位置

```
// 请记得按照备注部分描述初始化您的客户端
Awareness.SnapshotApi.getLocation(client)
    .setResultCallback(new ResultCallback<LocationResult>() {
        @Override
        public void onResult(@NonNull LocationResult locationResult) {
            Location location = locationResult.getLocation();
            Log.i(getClass().getSimpleName(), "坐标: " + location.getLatitude() + ", " +
                  location.getLongitude() + ", 半径 : " + location.getAccuracy());
        }
    });
});
```

第238.3节：使用Fence API获取用户活动变化

如果您想检测用户何时开始或结束诸如步行、跑步或任何其他DetectedActivityFence类中的活动，您可以为想要检测的活动创建一个fence，并在用户开始/结束该活动时收到通知。通过使用BroadcastReceiver，您将收到包含该活动数据的Intent：

```
// 您自己的动作过滤器，类似于Manifest中使用的那些。
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID + ".FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "walkingFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// 请确保按照备注部分描述初始化您的客户端。
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 等等。

    // 0 是一个标准的活动请求代码，可以根据需要更改。
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // 创建围栏。
    AwarenessFence fence = DetectedActivityFence.during(DetectedActivityFence.WALKING);
    // 注册围栏以接收回调。
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
    .build())
    .setResultCallback(new ResultCallback<Status>() {
        @Override
        public void onResult(@NonNull Status status) {
            if (status.isSuccess()) {
                Log.i(FENCE_KEY, "Successfully registered.");
            } else {
        }
    }
}
```

```
        case FenceState.FALSE:
            Log.i(TAG, "User is not in location");
            break;
        case FenceState.UNKNOWN:
            Log.i(TAG, "User is doing something unknown");
            break;
    }
}
```

Section 238.2: Get current location using Snapshot API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getLocation(client)
    .setResultCallback(new ResultCallback<LocationResult>() {
        @Override
        public void onResult(@NonNull LocationResult locationResult) {
            Location location = locationResult.getLocation();
            Log.i(getClass().getSimpleName(), "Coordinates: " + location.getLatitude() + "," +
                  location.getLongitude() + ", radius : " + location.getAccuracy());
        }
    });
});
```

Section 238.3: Get changes in user activity with Fence API

If you want to detect when your user starts or finishes an activity such as walking, running, or any other activity of the [DetectedActivityFence](#) class, you can create a [fence](#) for the activity that you want to detect, and get notified when your user starts/finishes this activity. By using a `BroadcastReceiver`, you will get an `Intent` with data that contains the activity:

```
// Your own action filter, like the ones used in the Manifest.
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "walkingFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section.
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // etc.

    // The 0 is a standard Activity request code that can be changed to your needs.
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence.
    AwarenessFence fence = DetectedActivityFence.during(DetectedActivityFence.WALKING);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
        .setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(@NonNull Status status) {
                if (status.isSuccess()) {
                    Log.i(FENCE_KEY, "Successfully registered.");
                } else {

```

```
        Log.e(FENCE_KEY, "Could not be registered: " + status);
    }
}
});  
}  
}
```

现在你可以使用 `BroadcastReceiver` 接收意图，以便在用户活动发生变化时获得回调：

```
public class FenceReceiver extends BroadcastReceiver {  
  
    private static final String TAG = "FenceReceiver";  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // 获取围栏状态  
        FenceState fenceState = FenceState.extract(intent);  
  
        switch (fenceState.getCurrentState()) {  
            case FenceState.TRUE:  
                Log.i(TAG, "用户正在行走");  
                break;  
            case FenceState.FALSE:  
                Log.i(TAG, "用户未在行走");  
                break;  
            case FenceState.UNKNOWN:  
                Log.i(TAG, "用户正在执行未知操作");  
                break;  
        }  
    }  
}
```

第238.4节：使用快照API获取当前用户活动

对于一次性、非持续的用户身体活动请求，请使用快照API：

```
// 请记得按照备注部分描述初始化你的客户端
Awareness.SnapshotApi.getDetectedActivity(client)
    .setResultCallback(new ResultCallback<DetectedActivityResult>() {
        @Override
        public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
            if (!detectedActivityResult.getStatus().isSuccess()) {
                Log.e(getClass().getSimpleName(), "无法获取当前活动。");
                return;
            }
            ActivityRecognitionResult result = detectedActivityResult
                .getActivityRecognitionResult();
            DetectedActivity probableActivity = result.getMostProbableActivity();
            Log.i(getClass().getSimpleName(), "接收到的活动：" +
probableActivity.toString());
        }
    });
});
```

第238.5节：使用Snapshot API获取耳机状态

```
// 请记得按照备注部分描述初始化你的客户端
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
```

```
        Log.e(FENCE_KEY, "Could not be registered: " + status);
    }
}
});  
}
```

Now you can receive the intent with a `BroadcastReceiver` to get callbacks when the user changes the activity:

```
public class FenceReceiver extends BroadcastReceiver {  
  
    private static final String TAG = "FenceReceiver";  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        // Get the fence state  
        FenceState fenceState = FenceState.extract(intent);  
  
        switch (fenceState.getCurrentState()) {  
            case FenceState.TRUE:  
                Log.i(TAG, "User is walking");  
                break;  
            case FenceState.FALSE:  
                Log.i(TAG, "User is not walking");  
                break;  
            case FenceState.UNKNOWN:  
                Log.i(TAG, "User is doing something unknown");  
                break;  
        }  
    }  
}
```

Section 238.4: Get current user activity using Snapshot API

For one-time, non-constant requests for a user's physical activity, use the Snapshot API:

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getDetectedActivity(client)
    .setResultCallback(new ResultCallback<DetectedActivityResult>() {
        @Override
        public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
            if (!detectedActivityResult.getStatus().isSuccess()) {
                Log.e(getClass().getSimpleName(), "Could not get the current activity.");
                return;
            }
            ActivityRecognitionResult result = detectedActivityResult
                .getActivityRecognitionResult();
            DetectedActivity probableActivity = result.getMostProbableActivity();
            Log.i(getClass().getSimpleName(), "Activity received : " +
                probableActivity.toString());
        }
    });
});
```

Section 238.5: Get headphone state with Snapshot API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
```

```

public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
    Log.i(TAG, "耳机状态连接状态: " +
        headphoneStateResult.getHeadphoneState()
        .getState() == HeadphoneState.PLUGGED_IN);
}
});

```

第238.6节：使用Snapshot API获取附近地点

```

// 请记得按照备注部分描述初始化你的客户端
Awareness.SnapshotApi.getPlaces(client)
    .setResultCallback(new ResultCallback<PlacesResult>() {
        @Override
        public void onResult(@NonNull PlacesResult placesResult) {
            List<PlaceLikelihood> likelihoodList = placesResult.getPlaceLikelihoods();
            if (likelihoodList == null || likelihoodList.isEmpty()) {
                Log.e(getClass().getSimpleName(), "没有可能的地点");
            }
        }
    });

```

关于获取这些地点中的数据，以下是一些选项：

```

Place place = placeLikelihood.getPlace();
String likelihood = placeLikelihood.getLikelihood();
Place place = likelihood.getPlace();
String placeName = place.getName();
String placeAddress = place.getAddress();
String placeCoords = place.getLatLang();
String locale = extractFromLocale(place.getLocale());

```

第238.7节：使用Snapshot API获取当前天气

```

// 请记得按照备注部分描述初始化你的客户端
Awareness.SnapshotApi.getWeather(client)
    .setResultCallback(new ResultCallback<WeatherResult>() {
        @Override
        public void onResult(@NonNull WeatherResult weatherResult) {
            Weather weather = weatherResult.getWeather();
            if (weather == null) {
                Log.e(getClass().getSimpleName(), "未接收到天气信息");
            } else {
                Log.i(getClass().getSimpleName(), "温度是 " +
                    weather.getTemperature(Weather.CELSIUS) + ", 体感温度是 " +
                    weather.getFeelsLikeTemperature(Weather.CELSIUS) +
                    ", 湿度是 " + weather.getHumidity());
            }
        }
    });

```

```

public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
    Log.i(TAG, "Headphone state connection state: " +
        headphoneStateResult.getHeadphoneState()
        .getState() == HeadphoneState.PLUGGED_IN));
}
);

```

Section 238.6: Get nearby places using Snapshot API

```

// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getPlaces(client)
    .setResultCallback(new ResultCallback<PlacesResult>() {
        @Override
        public void onResult(@NonNull PlacesResult placesResult) {
            List<PlaceLikelihood> likelihoodList = placesResult.getPlaceLikelihoods();
            if (likelihoodList == null || likelihoodList.isEmpty()) {
                Log.e(getClass().getSimpleName(), "No likely places");
            }
        }
    });

```

As for getting the data in those places, here are some options:

```

Place place = placeLikelihood.getPlace();
String likelihood = placeLikelihood.getLikelihood();
Place place = likelihood.getPlace();
String placeName = place.getName();
String placeAddress = place.getAddress();
String placeCoords = place.getLatLang();
String locale = extractFromLocale(place.getLocale());

```

Section 238.7: Get current weather using Snapshot API

```

// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getWeather(client)
    .setResultCallback(new ResultCallback<WeatherResult>() {
        @Override
        public void onResult(@NonNull WeatherResult weatherResult) {
            Weather weather = weatherResult.getWeather();
            if (weather == null) {
                Log.e(getClass().getSimpleName(), "No weather received");
            } else {
                Log.i(getClass().getSimpleName(), "Temperature is " +
                    weather.getTemperature(Weather.CELSIUS) + ", feels like " +
                    weather.getFeelsLikeTemperature(Weather.CELSIUS) +
                    ", humidity is " + weather.getHumidity());
            }
        }
    });

```

第239章：适用于

Android的Google地图API v2

参数

Google地图 Google地图是一个在onMapReady()事件中接收的对象

MarkerOptions MarkerOptions是Marker的构建类，用于向地图添加一个标记。

详细信息

Chapter 239: Google Maps API v2 for Android

Parameter

GoogleMap the GoogleMap is an object that is received on a onMapReady() event

MarkerOptions MarkerOptions is the builder class of a Marker, and is used to add one marker to a map.

Details

第239.1节：自定义Google地图样式

地图样式

Google 地图附带一组可应用的不同样式，使用以下代码：

```
// 设置地图类型为“混合”
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

不同的地图样式有：

普通

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

典型的道路地图。显示道路、一些人工特征以及重要的自然特征，如河流。道路和特征标签也可见。



Section 239.1: Custom Google Map Styles

Map Style

Google Maps come with a set of different styles to be applied, using this code :

```
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

The different map styles are :

Normal

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

Typical road map. Roads, some man-made features, and important natural features such as rivers are shown. Road and feature labels are also visible.



混合

```
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

卫星照片数据加上道路地图。道路和特征标签也可见。



Hybrid

```
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Satellite photograph data with road maps added. Road and feature labels are also visible.



卫星

```
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

卫星照片数据。道路和地物标签不可见。

Satellite

```
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

Satellite photograph data. Road and feature labels are not visible.



地形

```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

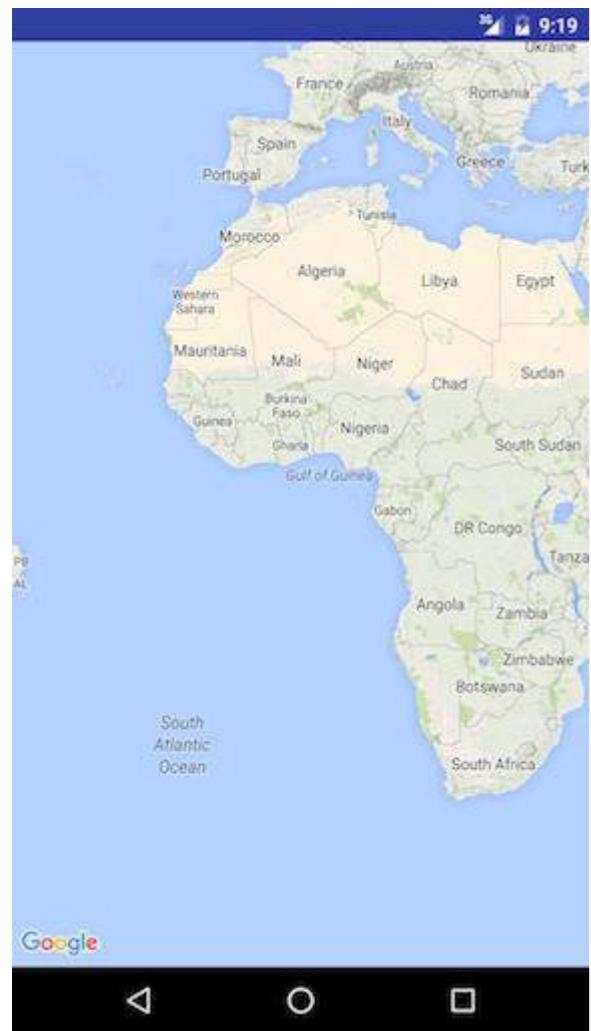
地形数据。地图包含颜色、等高线和标签，以及透视阴影。部分道路和标签也可见。



Terrain

```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Topographic data. The map includes colors, contour lines and labels, and perspective shading. Some roads and labels are also visible.



无

```
map.setMapType(GoogleMap.MAP_TYPE_NONE);
```

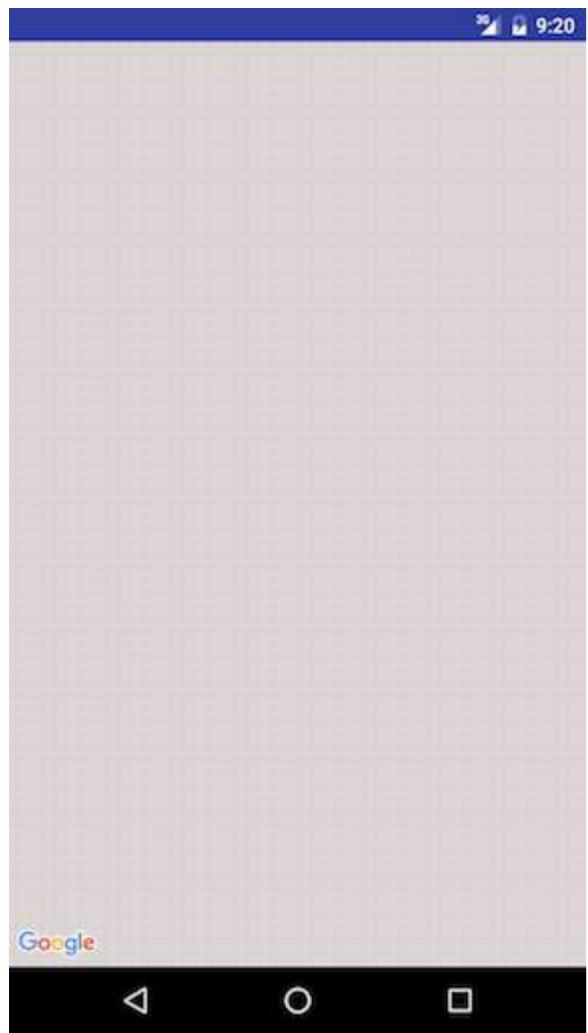
无图块。地图将以空白网格渲染，且不加载任何图块。



None

```
map.setMapType(GoogleMap.MAP_TYPE_NONE);
```

No tiles. The map will be rendered as an empty grid with no tiles loaded.



其他样式选项

室内地图

在高缩放级别下，地图将显示室内空间的楼层平面图。这些称为室内地图，仅在“普通”和“卫星”地图类型中显示。

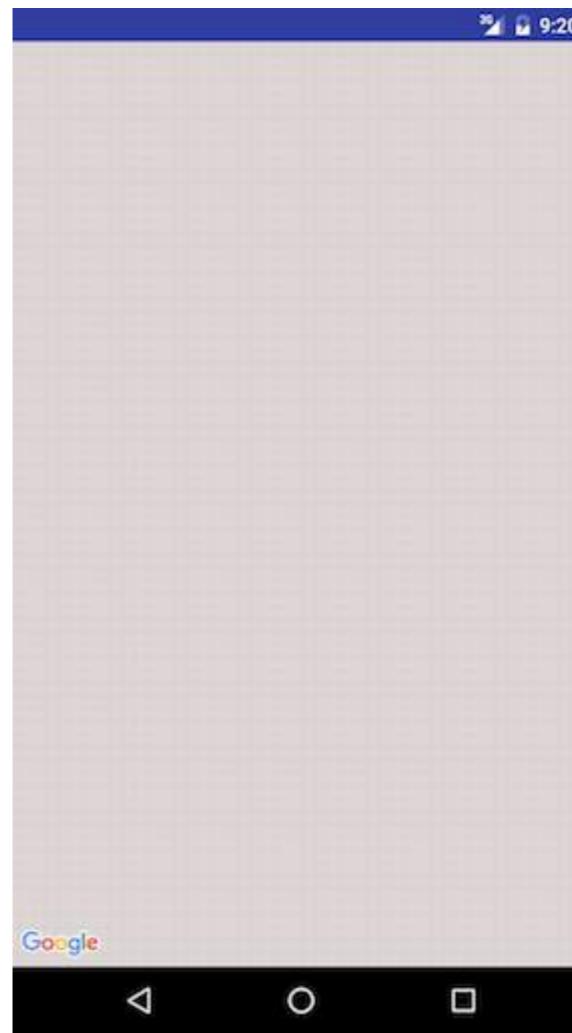
启用或禁用室内地图的方法如下：

```
GoogleMap.setIndoorEnabled(true);  
GoogleMap.setIndoorEnabled(false);
```

我们可以为地图添加自定义样式。

在 `onMapReady` 方法中添加以下代码片段

```
mMap = googleMap;  
try {  
    // 使用定义在原始资源文件中的 JSON 对象自定义基础地图的样式。  
  
    boolean success = mMap.setMapStyle(  
        MapStyleOptions.loadRawResourceStyle(  
            MapsActivity.this, R.raw.style_json));  
  
    if (!success) {  
        Log.e(TAG, "样式解析失败。");  
    }  
} catch (Resources.NotFoundException e) {
```



OTHER STYLE OPTIONS

Indoor Maps

At high zoom levels, the map will show floor plans for indoor spaces. These are called indoor maps, and are displayed only for the 'normal' and 'satellite' map types.

to enable or disable indoor maps, this is how it's done :

```
GoogleMap.setIndoorEnabled(true);  
GoogleMap.setIndoorEnabled(false);
```

We can add custom styles to maps.

In `onMapReady` method add the following code snippet

```
mMap = googleMap;  
try {  
    // Customise the styling of the base map using a JSON object defined  
    // in a raw resource file.  
    boolean success = mMap.setMapStyle(  
        MapStyleOptions.loadRawResourceStyle(  
            MapsActivity.this, R.raw.style_json));  
  
    if (!success) {  
        Log.e(TAG, "Style parsing failed.");  
    }  
} catch (Resources.NotFoundException e) {
```

```
Log.e(TAG, "找不到样式。", e);  
}
```

在 res 文件夹下创建一个名为 raw 的文件夹，并添加样式的 json 文件。示例 style.json 文件

```
[  
{  
    "featureType": "all",  
    "elementType": "geometry",  
    "stylers": [  
        {  
            "color": "#242f3e"  
        }  
    ]  
},  
{  
    "featureType": "all",  
    "elementType": "labels.text.stroke",  
    "stylers": [  
        {  
            "lightness": -80  
        }  
    ]  
},  
{  
    "featureType": "行政区划",  
    "elementType": "标签.文字填充",  
    "stylers": [  
        {  
            "color": "#746855"  
        }  
    ]  
},  
{  
    "featureType": "行政区划.地方",  
    "elementType": "标签.文字填充",  
    "stylers": [  
        {  
            "color": "#d59563"  
        }  
    ]  
},  
{  
    "featureType": "兴趣点",  
    "elementType": "标签.文字填充",  
    "stylers": [  
        {  
            "color": "#d59563"  
        }  
    ]  
},  
{  
    "featureType": "兴趣点.公园",  
    "elementType": "几何形状",  
    "stylers": [  
        {  
            "color": "#263c3f"  
        }  
    ]  
},  
]
```

```
Log.e(TAG, "Can't find style.", e);  
}
```

under res folder create a folder name raw and add the styles json file. Sample style.json file

```
[  
{  
    "featureType": "all",  
    "elementType": "geometry",  
    "stylers": [  
        {  
            "color": "#242f3e"  
        }  
    ]  
},  
{  
    "featureType": "all",  
    "elementType": "labels.text.stroke",  
    "stylers": [  
        {  
            "lightness": -80  
        }  
    ]  
},  
{  
    "featureType": "administrative",  
    "elementType": "labels.text.fill",  
    "stylers": [  
        {  
            "color": "#746855"  
        }  
    ]  
},  
{  
    "featureType": "administrative.locality",  
    "elementType": "labels.text.fill",  
    "stylers": [  
        {  
            "color": "#d59563"  
        }  
    ]  
},  
{  
    "featureType": "poi",  
    "elementType": "labels.text.fill",  
    "stylers": [  
        {  
            "color": "#d59563"  
        }  
    ]  
},  
{  
    "featureType": "poi.park",  
    "elementType": "geometry",  
    "stylers": [  
        {  
            "color": "#263c3f"  
        }  
    ]  
},  
]
```

```
"featureType": "poi.park",
"elementType": "labels.text.fill",
"stylers": [
  {
    "color": "#6b9a76"
  }
],
{
  "featureType": "road",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#2b3544"
    }
  ]
},
{
  "featureType": "road",
  "elementType": "labels.text.fill",
  "stylers": [
    {
      "color": "#9ca5b3"
    }
  ],
  {
    "featureType": "road.arterial",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#38414e"
      }
    ],
    {
      "featureType": "road.arterial",
      "elementType": "geometry.stroke",
      "stylers": [
        {
          "color": "#212a37"
        }
      ],
      {
        "featureType": "road.highway",
        "elementType": "geometry.fill",
        "stylers": [
          {
            "color": "#746855"
          }
        ],
        {
          "featureType": "road.highway",
          "elementType": "geometry.stroke",
          "stylers": [
            {
              "color": "#1f2835"
            }
          ]
        }
      ]
    }
  }
]
```

```
"featureType": "poi.park",
"elementType": "labels.text.fill",
"stylers": [
  {
    "color": "#6b9a76"
  }
],
{
  "featureType": "road",
  "elementType": "geometry.fill",
  "stylers": [
    {
      "color": "#2b3544"
    }
  ],
  {
    "featureType": "road",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#9ca5b3"
      }
    ],
    {
      "featureType": "road.arterial",
      "elementType": "geometry.fill",
      "stylers": [
        {
          "color": "#38414e"
        }
      ],
      {
        "featureType": "road.arterial",
        "elementType": "geometry.stroke",
        "stylers": [
          {
            "color": "#212a37"
          }
        ],
        {
          "featureType": "road.highway",
          "elementType": "geometry.fill",
          "stylers": [
            {
              "color": "#746855"
            }
          ],
          {
            "featureType": "road.highway",
            "elementType": "geometry.stroke",
            "stylers": [
              {
                "color": "#1f2835"
              }
            ]
          }
        }
      ]
    }
  }
]
```

```
{  
  "featureType": "road.highway",  
  "elementType": "labels.text.fill",  
  "stylers": [  
    {  
      "color": "#f3d19c"  
    }  
  ],  
  {  
    "featureType": "road.local",  
    "elementType": "geometry.fill",  
    "stylers": [  
      {  
        "color": "#38414e"  
      }  
    ]  
  },  
  {  
    "featureType": "road.local",  
    "elementType": "geometry.stroke",  
    "stylers": [  
      {  
        "color": "#212a37"  
      }  
    ]  
  },  
  {  
    "featureType": "transit",  
    "elementType": "geometry",  
    "stylers": [  
      {  
        "color": "#2f3948"  
      }  
    ]  
  },  
  {  
    "featureType": "transit.station",  
    "elementType": "labels.text.fill",  
    "stylers": [  
      {  
        "color": "#d59563"  
      }  
    ]  
  },  
  {  
    "featureType": "water",  
    "elementType": "geometry",  
    "stylers": [  
      {  
        "color": "#17263c"  
      }  
    ]  
  },  
  {  
    "featureType": "water",  
    "elementType": "labels.text.fill",  
    "stylers": [  
      {  
        "color": "#515c6d"  
      }  
    ]  
  }  
}
```

```
{  
  "featureType": "road.highway",  
  "elementType": "labels.text.fill",  
  "stylers": [  
    {  
      "color": "#f3d19c"  
    }  
  ],  
  {  
    "featureType": "road.local",  
    "elementType": "geometry.fill",  
    "stylers": [  
      {  
        "color": "#38414e"  
      }  
    ]  
  },  
  {  
    "featureType": "road.local",  
    "elementType": "geometry.stroke",  
    "stylers": [  
      {  
        "color": "#212a37"  
      }  
    ]  
  },  
  {  
    "featureType": "transit",  
    "elementType": "geometry",  
    "stylers": [  
      {  
        "color": "#2f3948"  
      }  
    ]  
  },  
  {  
    "featureType": "transit.station",  
    "elementType": "labels.text.fill",  
    "stylers": [  
      {  
        "color": "#d59563"  
      }  
    ]  
  },  
  {  
    "featureType": "water",  
    "elementType": "geometry",  
    "stylers": [  
      {  
        "color": "#17263c"  
      }  
    ]  
  },  
  {  
    "featureType": "water",  
    "elementType": "labels.text.fill",  
    "stylers": [  
      {  
        "color": "#515c6d"  
      }  
    ]  
  }  
}
```

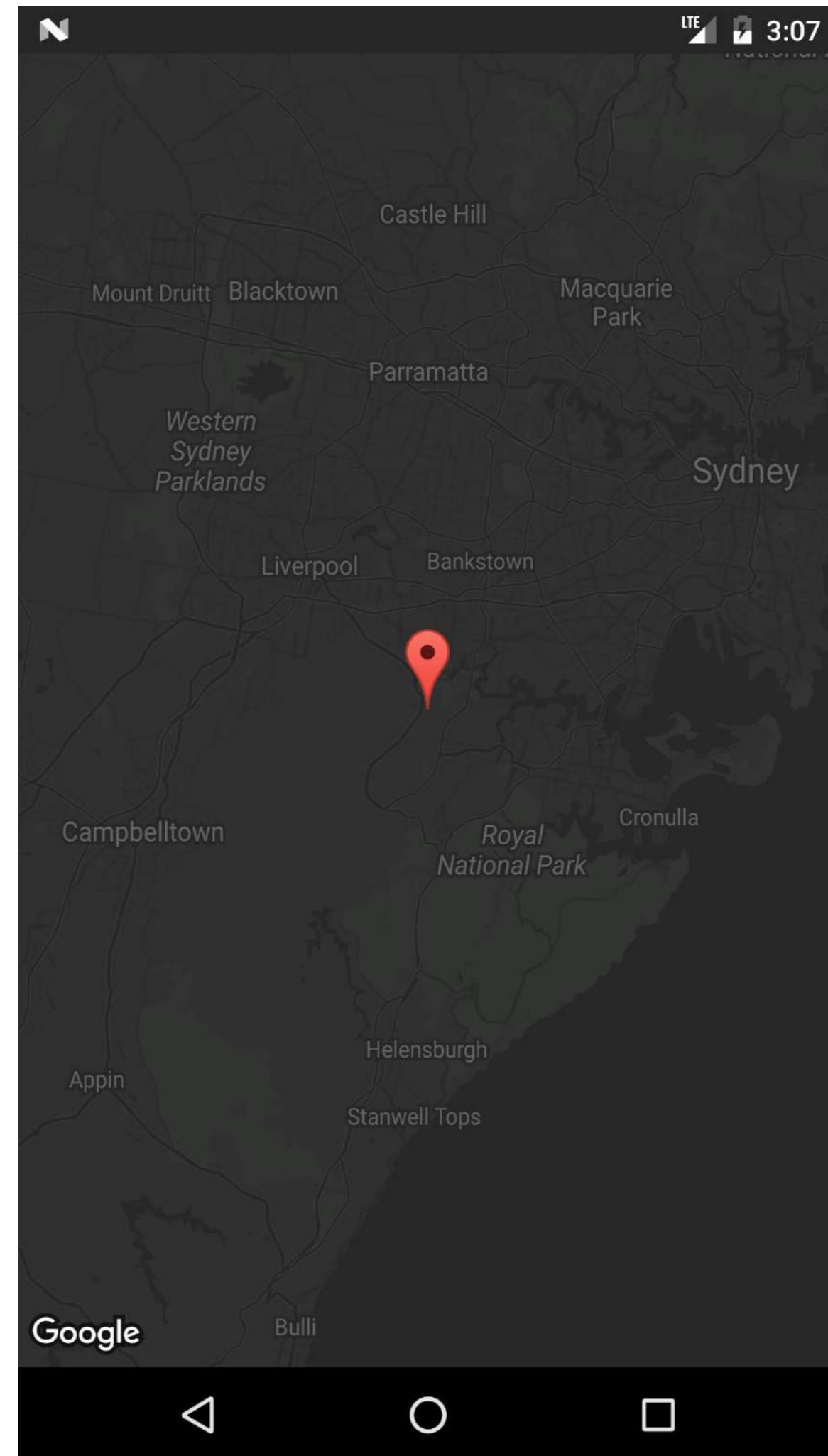
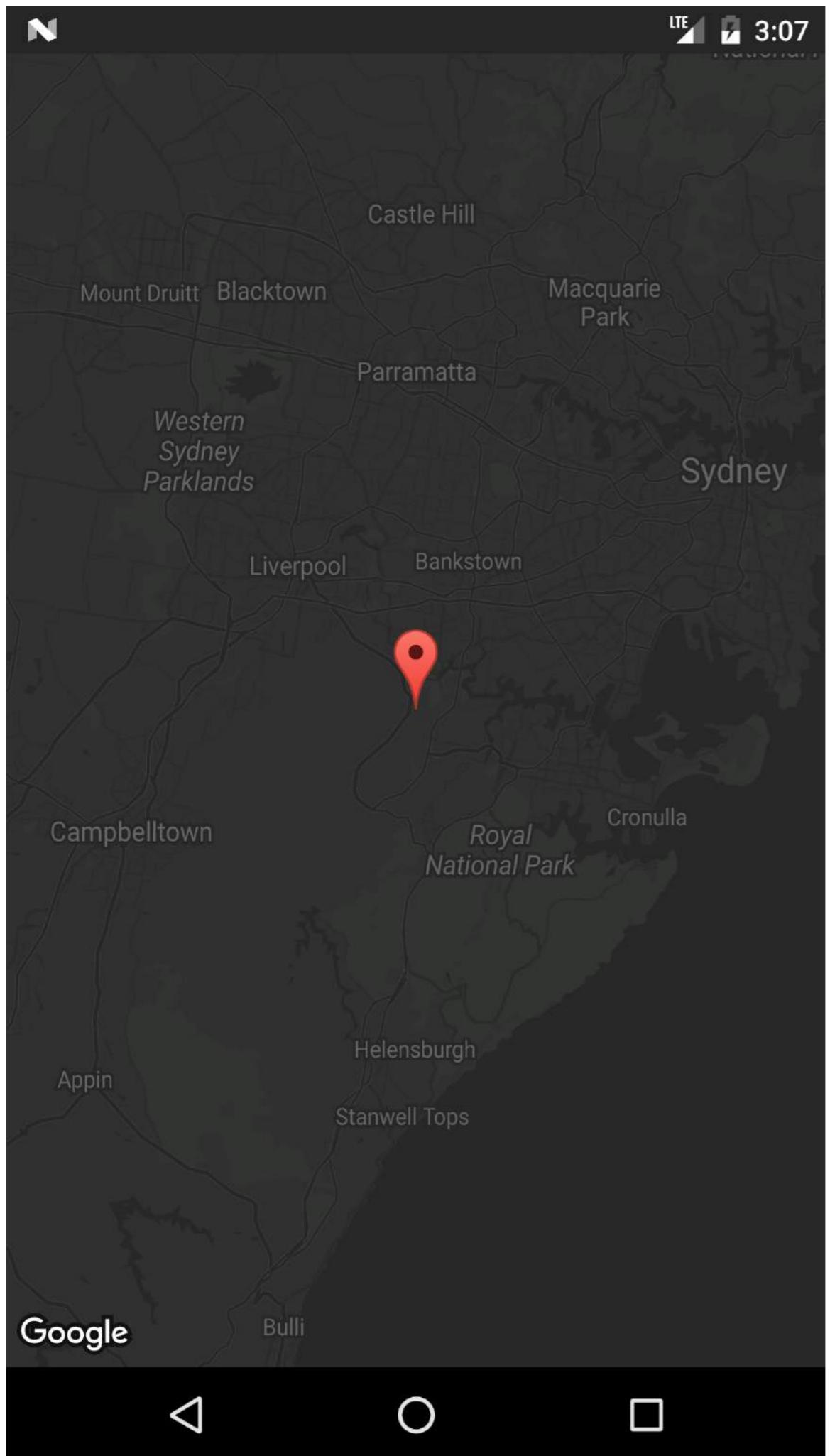
```
},
{
  "featureType": "water",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -20
    }
  ]
}
```

点击此链接生成样式json文件



```
},
{
  "featureType": "water",
  "elementType": "labels.text.stroke",
  "stylers": [
    {
      "lightness": -20
    }
  ]
}
```

To generate styles json file click this [link](#)



第239.2节：默认谷歌地图活动

此活动代码将提供使用SupportMapFragment包含谷歌地图的基本功能。

谷歌地图V2 API包含了一种全新的加载地图方式。

活动现在必须实现OnMapReadyCallBack接口，该接口带有一个onMapReady()方法重写，每次运行SupportMapFragment.getMapAsync(OnMapReadyCallback)且调用成功完成时都会执行该方法。

地图使用标记(Markers)、多边形(Polygons)和折线(PolyLines)向用户显示交互信息。

MapsActivity.java :

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback {  
  
    private GoogleMap mMap;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_maps);  
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
            .findFragmentById(R.id.map);  
        mapFragment.getMapAsync(this);  
    }  
  
    @Override  
    public void onMapReady(GoogleMap googleMap) {  
        mMap = googleMap;  
  
        // 在澳大利亚悉尼添加一个标记，并移动摄像头。  
        LatLng sydney = new LatLng(-34, 151);  
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));  
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
    }  
}
```

注意，上述代码加载了一个布局，该布局中嵌套了一个 SupportMapFragment，位于容器布局内，其 ID 定义为 R.id.map。布局文件如下所示：

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:tools="http://schemas.android.com/tools"  
        xmlns:map="http://schemas.android.com/apk/res-auto"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:id="@+id/map"  
        tools:context="com.example.app.MapsActivity"  
        android:name="com.google.android.gms.maps.SupportMapFragment"/>  
  
</LinearLayout>
```

Section 239.2: Default Google Map Activity

This Activity code will provide basic functionality for including a Google Map using a SupportMapFragment.

The Google Maps V2 API includes an all-new way to load maps.

Activities now have to implement the **OnMapReadyCallBack** interface, which comes with a **onMapReady()** method override that is executed every time we run **SupportMapFragment.getMapAsync(OnMapReadyCallback)**; and the call is successfully completed.

Maps use **Markers**, **Polygons** and **PolyLines** to show interactive information to the user.

MapsActivity.java:

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback {  
  
    private GoogleMap mMap;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_maps);  
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
            .findFragmentById(R.id.map);  
        mapFragment.getMapAsync(this);  
    }  
  
    @Override  
    public void onMapReady(GoogleMap googleMap) {  
        mMap = googleMap;  
  
        // Add a marker in Sydney, Australia, and move the camera.  
        LatLng sydney = new LatLng(-34, 151);  
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));  
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
    }  
}
```

Notice that the code above inflates a layout, which has a SupportMapFragment nested inside the container Layout, defined with an ID of R.id.map. The layout file is shown below:

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:tools="http://schemas.android.com/tools"  
        xmlns:map="http://schemas.android.com/apk/res-auto"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:id="@+id/map"  
        tools:context="com.example.app.MapsActivity"  
        android:name="com.google.android.gms.maps.SupportMapFragment"/>  
  
</LinearLayout>
```

第239.3节：在谷歌地图中显示当前位置

这是一个完整的Activity类，它在当前位置放置一个标记，并将摄像头移动到当前位置。

这里按顺序进行几个操作：

- 检查位置权限
- 一旦位置权限被授予，调用setMyLocationEnabled()，构建GoogleApiClient，并连接它
- 一旦GoogleApiClient连接成功，请求位置更新

```
public class MapLocationActivity extends AppCompatActivity
    implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    GoogleMap mGoogleMap;
    SupportMapFragment mapFrag;
    LocationRequest mLocationRequest;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    Marker mCurrLocationMarker;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getSupportActionBar().setTitle("地图定位活动");

        mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFrag.getMapAsync(this);
    }

    @Override
    public void onPause() {
        super.onPause();

        //当活动不再活跃时停止位置更新
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap)
    {
        mGoogleMap=googleMap;
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

        //初始化 Google Play 服务
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
                //位置权限已授予
                buildGoogleApiClient();
                mGoogleMap.setMyLocationEnabled(true);
            } else {
        
```

Section 239.3: Show Current Location in a Google Map

Here is a full Activity class that places a Marker at the current location, and also moves the camera to the current position.

There are a few things going on in sequence here:

- Check Location permission
 - Once Location permission is granted, call setMyLocationEnabled(), build the GoogleApiClient, and connect it
 - Once the GoogleApiClient is connected, request location updates
- ```
public class MapLocationActivity extends AppCompatActivity
 implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks,
 GoogleApiClient.OnConnectionFailedListener,
 LocationListener {

 GoogleMap mGoogleMap;
 SupportMapFragment mapFrag;
 LocationRequest mLocationRequest;
 GoogleApiClient mGoogleApiClient;
 Location mLastLocation;
 Marker mCurrLocationMarker;

 @Override
 protected void onCreate(Bundle savedInstanceState)
 {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);

 getSupportActionBar().setTitle("Map Location Activity");

 mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
 mapFrag.getMapAsync(this);
 }

 @Override
 public void onPause() {
 super.onPause();

 //stop location updates when Activity is no longer active
 if (mGoogleApiClient != null) {
 LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
 }
 }

 @Override
 public void onMapReady(GoogleMap googleMap)
 {
 mGoogleMap=googleMap;
 mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

 //Initialize Google Play Services
 if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
 if (ContextCompat.checkSelfPermission(this,
 Manifest.permission.ACCESS_FINE_LOCATION)
 == PackageManager.PERMISSION_GRANTED) {
 //Location Permission already granted
 buildGoogleApiClient();
 mGoogleMap.setMyLocationEnabled(true);
 } else {
```

```

 //请求位置权限
checkLocationPermission();
}

else {
buildGoogleApiClient();
 mGoogleMap.setMyLocationEnabled(true);
}

}

protected synchronized void buildGoogleApiClient() {
 mGoogleApiClient = new GoogleApiClient.Builder(this)
 .addConnectionCallbacks(this)
.addOnConnectionFailedListener(this)
 .addApi(LocationServices.API)
.build();
 mGoogleApiClient.connect();
}

@Override
public void onConnected(Bundle bundle) {
 mLocationRequest = new LocationRequest();
 mLocationRequest.setInterval(1000);
mLocationRequest.setFastestInterval(1000);
 mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
 if (ContextCompat.checkSelfPermission(this,
 Manifest.permission.ACCESS_FINE_LOCATION)
 == PackageManager.PERMISSION_GRANTED) {
LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
 }
}

@Override
public void onConnectionSuspended(int i) {}

@Override
public void onConnectionFailed(ConnectionResult connectionResult) {}

@Override
public void onLocationChanged(Location location)
{
mLastLocation = location;
 if (mCurrLocationMarker != null) {
 mCurrLocationMarker.remove();
 }

 //放置当前位置标记
LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
 MarkerOptions markerOptions = new MarkerOptions();
markerOptions.position(latLng);
markerOptions.title("当前位置");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

 //移动地图摄像头
mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
 mGoogleMap.animateCamera(CameraUpdateFactory.zoomTo(11));

 //停止位置更新
 if (mGoogleApiClient != null) {

```

```

 //Request Location Permission
checkLocationPermission();
}

else {
buildGoogleApiClient();
 mGoogleMap.setMyLocationEnabled(true);
}

}

protected synchronized void buildGoogleApiClient() {
 mGoogleApiClient = new GoogleApiClient.Builder(this)
 .addConnectionCallbacks(this)
.addOnConnectionFailedListener(this)
 .addApi(LocationServices.API)
.build();
 mGoogleApiClient.connect();
}

@Override
public void onConnected(Bundle bundle) {
 mLocationRequest = new LocationRequest();
 mLocationRequest.setInterval(1000);
mLocationRequest.setFastestInterval(1000);
 mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
 if (ContextCompat.checkSelfPermission(this,
 Manifest.permission.ACCESS_FINE_LOCATION)
 == PackageManager.PERMISSION_GRANTED) {
LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
 }
}

@Override
public void onConnectionSuspended(int i) {}

@Override
public void onConnectionFailed(ConnectionResult connectionResult) {}

@Override
public void onLocationChanged(Location location)
{
mLastLocation = location;
 if (mCurrLocationMarker != null) {
 mCurrLocationMarker.remove();
 }

 //Place current location marker
LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
 MarkerOptions markerOptions = new MarkerOptions();
markerOptions.position(latLng);
markerOptions.title("Current Position");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

 //move map camera
mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
 mGoogleMap.animateCamera(CameraUpdateFactory.zoomTo(11));

 //stop location updates
 if (mGoogleApiClient != null) {

```

```

LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
private void checkLocationPermission() {
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
 != PackageManager.PERMISSION_GRANTED) {

 // 我们是否应该显示解释?
 if (ActivityCompat.shouldShowRequestPermissionRationale(this,
 Manifest.permission.ACCESS_FINE_LOCATION)) {

 // 异步向用户显示说明——不要阻塞// 当前线程等待用户响应! 用户看到说明后, 尝试
 // 再次请求权限。
 new AlertDialog.Builder(this)
 .setTitle("需要位置权限")
 .setMessage("此应用需要位置权限, 请接受以使用
位置功能")
 .setPositiveButton("确定", new DialogInterface.OnClickListener() {
 @Override
 public void onClick(DialogInterface dialogInterface, int i) {
 // 显示说明后提示用户
 ActivityCompat.requestPermissions(MapLocationActivity.this,
 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
 MY_PERMISSIONS_REQUEST_LOCATION);
 }
 })
 .create()
 .show();

 } else {
 // 不需要说明, 我们可以直接请求权限。
 ActivityCompat.requestPermissions(this,
 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
 MY_PERMISSIONS_REQUEST_LOCATION);
 }
 }
}

@Override
public void onRequestPermissionsResult(int requestCode,
 String permissions[], int[] grantResults) {
 switch (requestCode) {
 case MY_PERMISSIONS_REQUEST_LOCATION: {
 // 如果请求被取消, 结果数组将为空。
 if (grantResults.length > 0
 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

 // 权限已被授予, 太好了! 执行你需要的
 // 与位置相关的任务。
 if (ContextCompat.checkSelfPermission(this,
 Manifest.permission.ACCESS_FINE_LOCATION)
 == PackageManager.PERMISSION_GRANTED) {

 if (mGoogleApiClient == null) {
 buildGoogleApiClient();
 }
 }
 mGoogleMap.setMyLocationEnabled(true);
 }
 }
}

```

```

LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
private void checkLocationPermission() {
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
 != PackageManager.PERMISSION_GRANTED) {

 // Should we show an explanation?
 if (ActivityCompat.shouldShowRequestPermissionRationale(this,
 Manifest.permission.ACCESS_FINE_LOCATION)) {

 // Show an explanation to the user *asynchronously* -- don't block
 // this thread waiting for the user's response! After the user
 // sees the explanation, try again to request the permission.
 new AlertDialog.Builder(this)
 .setTitle("Location Permission Needed")
 .setMessage("This app needs the Location permission, please accept to use
location functionality")
 .setPositiveButton("OK", new DialogInterface.OnClickListener() {
 @Override
 public void onClick(DialogInterface dialogInterface, int i) {
 //Prompt the user once explanation has been shown
 ActivityCompat.requestPermissions(MapLocationActivity.this,
 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
 MY_PERMISSIONS_REQUEST_LOCATION);
 }
 })
 .create()
 .show();

 } else {
 // No explanation needed, we can request the permission.
 ActivityCompat.requestPermissions(this,
 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
 MY_PERMISSIONS_REQUEST_LOCATION);
 }
 }
}

@Override
public void onRequestPermissionsResult(int requestCode,
 String permissions[], int[] grantResults) {
 switch (requestCode) {
 case MY_PERMISSIONS_REQUEST_LOCATION: {
 // If request is cancelled, the result arrays are empty.
 if (grantResults.length > 0
 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

 // permission was granted, yay! Do the
 // location-related task you need to do.
 if (ContextCompat.checkSelfPermission(this,
 Manifest.permission.ACCESS_FINE_LOCATION)
 == PackageManager.PERMISSION_GRANTED) {

 if (mGoogleApiClient == null) {
 buildGoogleApiClient();
 }
 }
 mGoogleMap.setMyLocationEnabled(true);
 }
 }
}

```

```

 } else {

 // 权限被拒绝，真遗憾！禁用
 // 依赖此权限的功能。
 Toast.makeText(this, "权限被拒绝", Toast.LENGTH_LONG).show();
 }
 return;
}

// 其他 'case' 行用于检查此应用可能请求的其他
// 权限
}
}

```

activity\_main.xml :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical" android:layout_width="match_parent"
 android:layout_height="match_parent">

 <fragment xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 xmlns:map="http://schemas.android.com/apk/res-auto"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:id="@+id/map"
 tools:context="com.example.app.MapLocationActivity"
 android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>

```

结果：

在 Marshmallow 和 Nougat 上如有需要，使用 AlertDialog 显示说明（这种情况发生在用户之前拒绝了权限请求，或者曾经授予权限后又在

设置中撤销了权限）：

```

 } else {

 // permission denied, boo! Disable the
 // functionality that depends on this permission.
 Toast.makeText(this, "permission denied", Toast.LENGTH_LONG).show();
 }
 return;
}

// other 'case' lines to check for other
// permissions this app might request
}
}

```

activity\_main.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical" android:layout_width="match_parent"
 android:layout_height="match_parent">

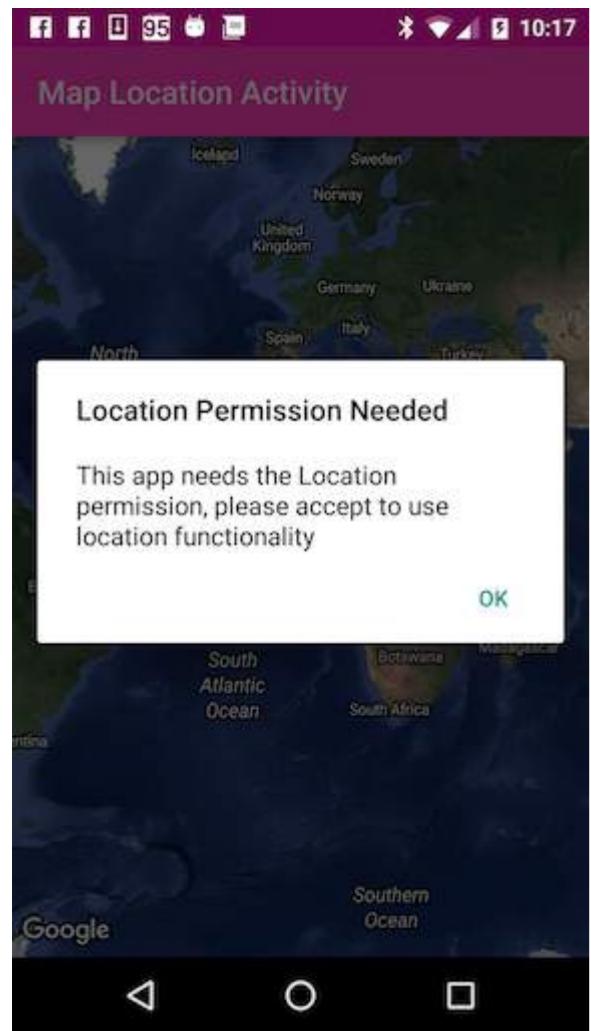
 <fragment xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 xmlns:map="http://schemas.android.com/apk/res-auto"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:id="@+id/map"
 tools:context="com.example.app.MapLocationActivity"
 android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>

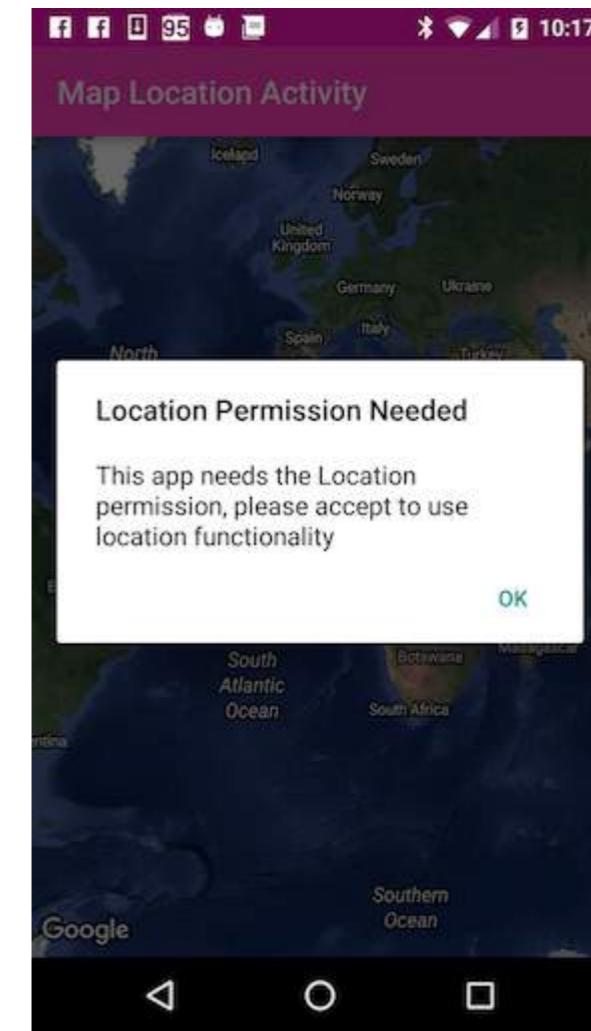
```

**Result:**

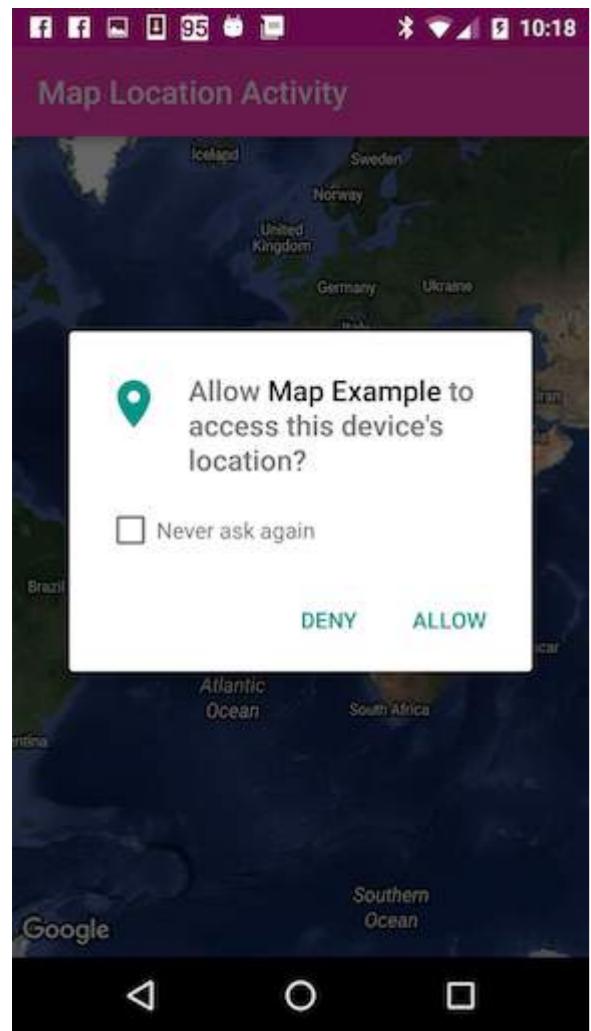
Show explanation if needed on Marshmallow and Nougat using an AlertDialog (this case happens when the user had previously denied a permission request, or had granted the permission and then later revoked it in the settings):



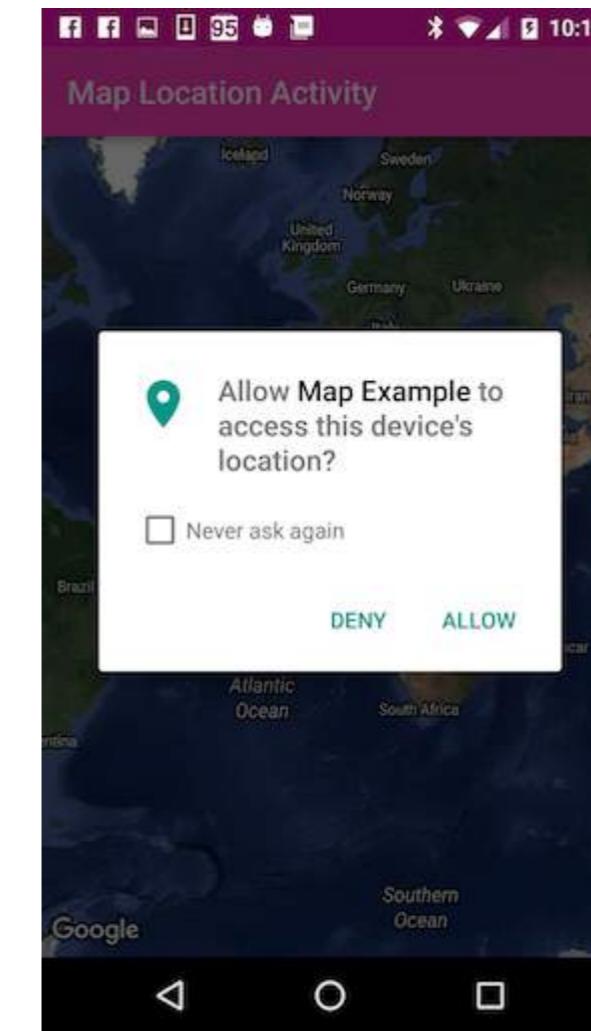
通过调用请求 Marshmallow 和 Nougat 上的定位权限  
`ActivityCompat.requestPermissions()` :



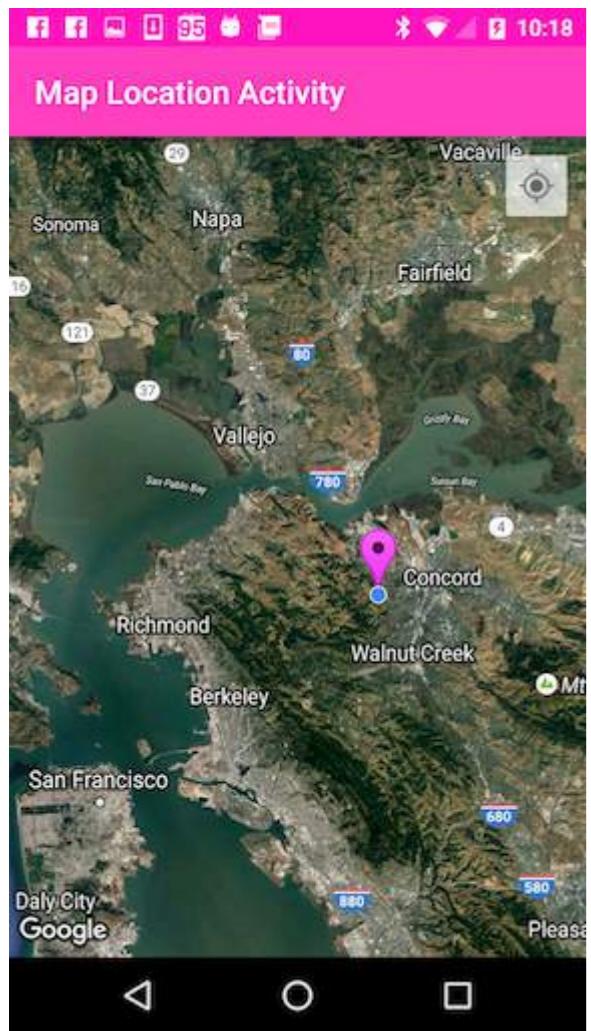
Prompt the user for Location permission on Marshmallow and Nougat by calling  
`ActivityCompat.requestPermissions()`:



当定位权限被授予时，将摄像头移动到当前位置并放置标记：



Move camera to current location and place Marker when the Location permission is granted:



## 第239.4节：更改偏移量

通过根据需要更改 mappoint x 和 y 的值，可以更改谷歌地图的偏移位置，默认情况下它会位于地图视图的中心。调用以下方法即可更改它！最好在你的 onLocationChanged 中使用，例如 changeOffsetCenter(location.getLatitude(),location.getLongitude());

```
public void changeOffsetCenter(double latitude, double longitude) {
 Point mappoint = mGoogleMap.getProjection().toScreenLocation(new LatLng(latitude,
 longitude));
 mappoint.set(mappoint.x, mappoint.y-100); // 根据需要更改这些值，当前为硬编码值，如果需要也可以基于比例如使用 DisplayMetrics 来设置

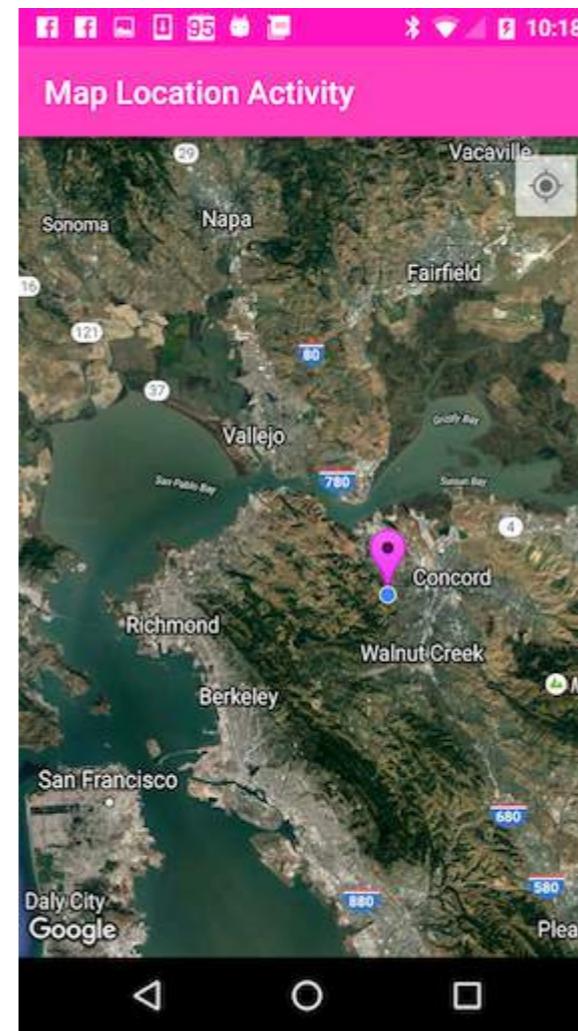
 mGoogleMap.animateCamera(CameraUpdateFactory.newLatLng(mGoogleMap.getProjection().fromScreenLocation(mappoint)));
}
```

## 第239.5节：MapView：在现有布局中嵌入 GoogleMap

如果使用提供的 MapView 类，可以将 GoogleMap 作为 Android 视图来处理。它的用法与 MapFragment 非常相似。

在你的布局中按如下方式使用 MapView：

```
<com.google.android.gms.maps.MapView
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:map="http://schemas.android.com/apk/res-auto"
```



## Section 239.4: Change Offset

By changing mappoint x and y values as you need you can change offset position of google map, by default it will be in the center of the map view. Call below method where you want to change it! Better to use it inside your onLocationChanged like changeOffsetCenter(location.getLatitude(),location.getLongitude());

```
public void changeOffsetCenter(double latitude, double longitude) {
 Point mappoint = mGoogleMap.getProjection().toScreenLocation(new LatLng(latitude,
 longitude));
 mappoint.set(mappoint.x, mappoint.y-100); // change these values as you need , just hard coded a value if you want you can give it based on a ratio like using DisplayMetrics as well

 mGoogleMap.animateCamera(CameraUpdateFactory.newLatLng(mGoogleMap.getProjection().fromScreenLocation(mappoint)));
}
```

## Section 239.5: MapView: embedding a GoogleMap in an existing layout

It is possible to treat a GoogleMap as an Android view if we make use of the provided MapView class. Its usage is very similar to MapFragment.

In your layout use MapView as follows:

```
<com.google.android.gms.maps.MapView
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:map="http://schemas.android.com/apk/res-auto"
```

```

 android:id="@+id/map"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
<!--
 map:mapType="0" 指定初始地图类型的更改map:zOrderOnTop="true" 控制地图视图的表面是否置于其窗口之上 map:useViewLifecycle="true" 当使用 MapFragment 时, 此标志指定地图的生命周期应绑定到片段的视图还是片段本身
 map:uiCompass="true" 启用或禁用指南针map:uiRotateGestures="true" 设置是否启用旋转手势的偏好
 map:uiScrollGestures="true" 设置是否启用滚动手势的偏好
 map:uiTiltGestures="true" 设置是否启用倾斜手势的偏好
 map:uiZoomGestures="true" 设置是否启用缩放手势的偏好
 map:uiZoomControls="true" 启用或禁用缩放控件
 map:liteMode="true" 指定地图是否应以精简模式创建
 map:uiMapToolbar="true" 指定是否启用地图工具栏
 map:ambientEnabled="true" 指定是否启用环境模式样式
 map:cameraMinZoomPreference="0.0" 指定摄像机缩放的首选下限
 map:cameraMaxZoomPreference="1.0" 指定摄像机缩放的首选上限 -->
/>

```

您的活动需要实现 OnMapReadyCallback 接口才能正常工作：

```

/**
 * 这展示了如何创建一个带有原始 MapView 的简单活动并向其添加标记。
 * 这需要将所有重要的生命周期方法转发给 MapView。
 */
public class RawMapViewDemoActivity extends AppCompatActivity implements OnMapReadyCallback {

 private MapView mMapView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.raw_mapview_demo);

 mMapView = (MapView) findViewById(R.id.map);
 mMapView.onCreate(savedInstanceState);

 mMapView.getMapAsync(this);
 }

 @Override
 protected void onResume() {
 super.onResume();
 mMapView.onResume();
 }

 @Override
 public void onMapReady(GoogleMap map) {
 map.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
 }

 @Override
 protected void onPause() {
 mMapView.onPause();
 super.onPause();
 }
}

```

```

 android:id="@+id/map"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
<!--
 map:mapType="0" Specifies a change to the initial map type
 map:zOrderOnTop="true" Control whether the map view's surface is placed on top of its window
 map:useViewLifecycle="true" When using a MapFragment, this flag specifies whether the lifecycle of the map should be tied to the fragment's view or the fragment itself
 map:uiCompass="true" Enables or disables the compass
 map:uiRotateGestures="true" Sets the preference for whether rotate gestures should be enabled or disabled
 map:uiScrollGestures="true" Sets the preference for whether scroll gestures should be enabled or disabled
 map:uiTiltGestures="true" Sets the preference for whether tilt gestures should be enabled or disabled
 map:uiZoomGestures="true" Sets the preference for whether zoom gestures should be enabled or disabled
 map:uiZoomControls="true" Enables or disables the zoom controls
 map:liteMode="true" Specifies whether the map should be created in lite mode
 map:uiMapToolbar="true" Specifies whether the mapToolbar should be enabled
 map:ambientEnabled="true" Specifies whether ambient-mode styling should be enabled
 map:cameraMinZoomPreference="0.0" Specifies a preferred lower bound for camera zoom
 map:cameraMaxZoomPreference="1.0" Specifies a preferred upper bound for camera zoom -->
/>

```

Your activity needs to implement the OnMapReadyCallback interface in order to work:

```

/**
 * This shows how to create a simple activity with a raw MapView and add a marker to it. This
 * requires forwarding all the important lifecycle methods onto MapView.
 */
public class RawMapViewDemoActivity extends AppCompatActivity implements OnMapReadyCallback {

 private MapView mMapView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.raw_mapview_demo);

 mMapView = (MapView) findViewById(R.id.map);
 mMapView.onCreate(savedInstanceState);

 mMapView.getMapAsync(this);
 }

 @Override
 protected void onResume() {
 super.onResume();
 mMapView.onResume();
 }

 @Override
 public void onMapReady(GoogleMap map) {
 map.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
 }

 @Override
 protected void onPause() {
 mMapView.onPause();
 super.onPause();
 }
}

```

```

}
@Override
protected void onDestroy() {
 mMapView.onDestroy();
 super.onDestroy();
}

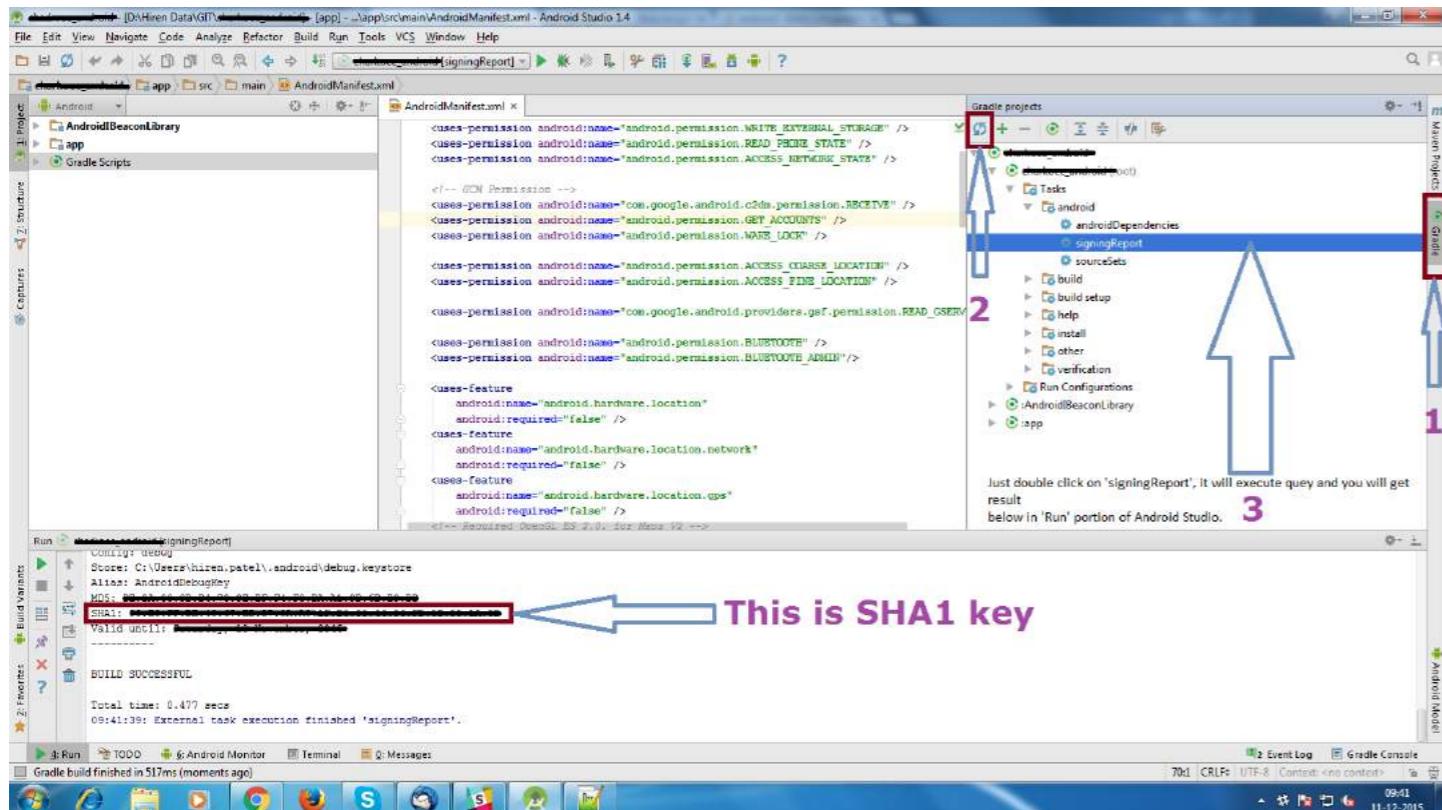
@Override
public void onLowMemory() {
 super.onLowMemory();
 mMapView.onLowMemory();
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 mMapView.onSaveInstanceState(outState);
}
}

```

## 第239.6节：获取调试用SHA1指纹

1. 打开Android Studio
2. 打开你的项目
3. 点击Gradle (在右侧面板，你会看到Gradle栏)
4. 点击刷新 (在Gradle栏点击刷新，你会看到项目的列表Gradle脚本)
5. 点击你的项目 (在列表中点击你的项目名称 (根目录))
6. 点击任务
7. 点击 android
8. 双击 signingReport (你将在运行栏中获得SHA1和MD5)



```

}
@Override
protected void onDestroy() {
 mMapView.onDestroy();
 super.onDestroy();
}

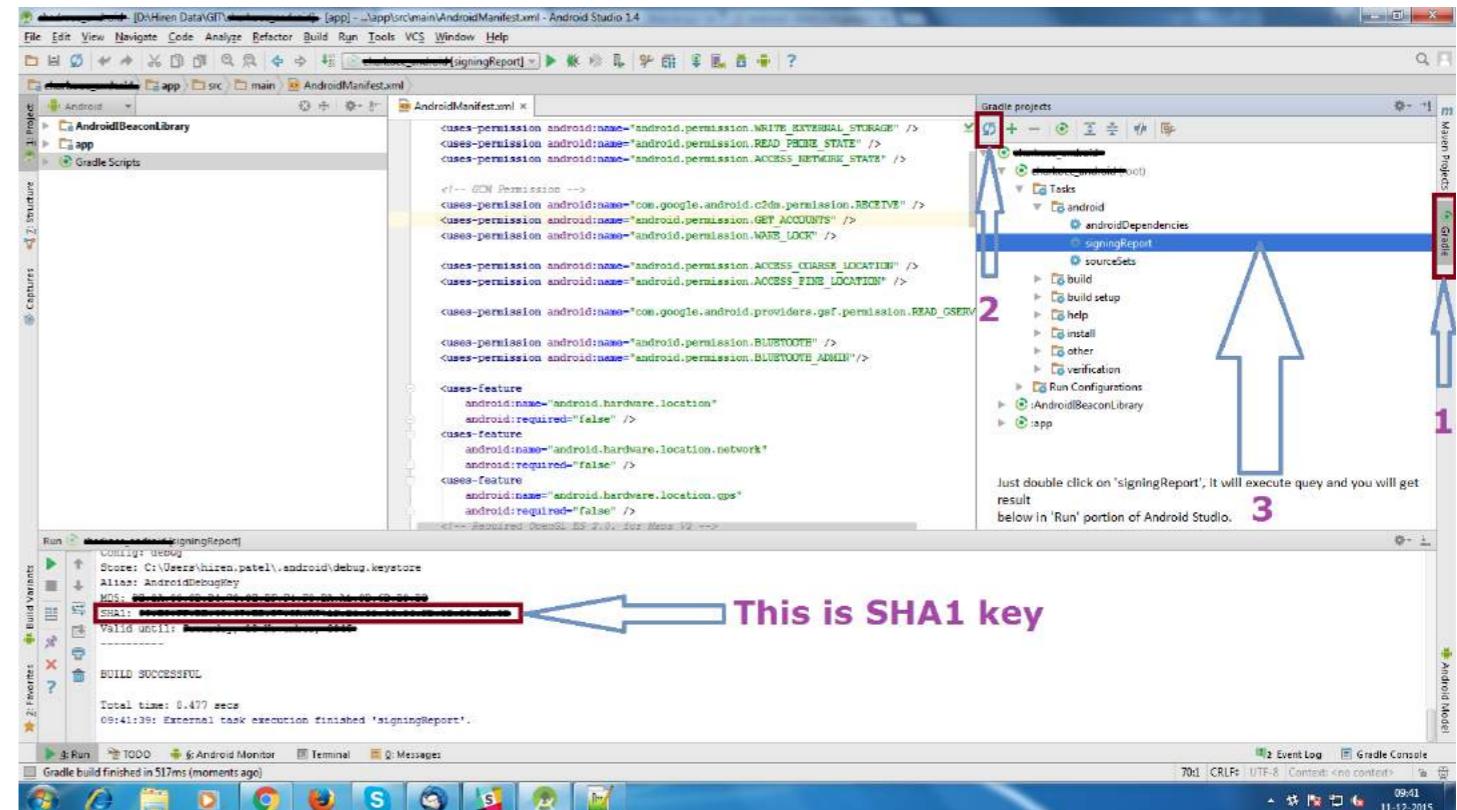
@Override
public void onLowMemory() {
 super.onLowMemory();
 mMapView.onLowMemory();
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 mMapView.onSaveInstanceState(outState);
}
}

```

## Section 239.6: Get debug SHA1 fingerprint

1. Open Android Studio
2. Open Your Project
3. Click on Gradle (From Right Side Panel, you will see **Gradle Bar**)
4. Click on Refresh (Click on Refresh from **Gradle Bar**, you will see **List** Gradle scripts of your Project)
5. Click on Your Project (Your Project Name form **List** (root))
6. Click on Tasks
7. Click on android
8. Double Click on signingReport (You will get **SHA1** and **MD5** in **Run Bar**)



## 第239.7节：向地图添加标记

例如，要从一个ArrayList的MyLocation对象中向谷歌地图添加标记，我们可以这样做。

MyLocation持有者类：

```
public class MyLocation {
 LatLng latLng;
 String title;
 String snippet;
}
```

这是一个方法，它将接受一个MyLocation对象列表，并为每个对象放置一个标记：

```
private void LocationsLoaded(List<MyLocation> locations){

 for (MyLocation myLoc : locations){
 mMap.addMarker(new MarkerOptions()
 .position(myLoc.latLng)
 .title(myLoc.title)
 .snippet(myLoc.snippet)
 .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 }
}
```

注意：在此示例中，mMap 是 Activity 的一个成员变量，我们已将其赋值为 onMapReady() 重写方法中接收到的地图引用。

## 第239.8节：UISettings

使用 [UISettings](#)，可以修改 Google 地图的外观。

以下是一些常用设置的示例：

```
mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
mGoogleMap.getUiSettings().setMapToolbarEnabled(true);
mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
mGoogleMap.getUiSettings().setCompassEnabled(true);
```

结果：

## Section 239.7: Adding markers to a map

To add markers to a Google Map, for example from an [ArrayList](#) of MyLocation Objects, we can do it this way.

The MyLocation holder class:

```
public class MyLocation {
 LatLng latLng;
 String title;
 String snippet;
}
```

Here is a method that would take a list of MyLocation Objects and place a Marker for each one:

```
private void LocationsLoaded(List<MyLocation> locations){

 for (MyLocation myLoc : locations){
 mMap.addMarker(new MarkerOptions()
 .position(myLoc.latLng)
 .title(myLoc.title)
 .snippet(myLoc.snippet)
 .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 }
}
```

Note: For the purpose of this example, mMap is a class member variable of the Activity, where we've assigned it to the map reference received in the onMapReady() override.

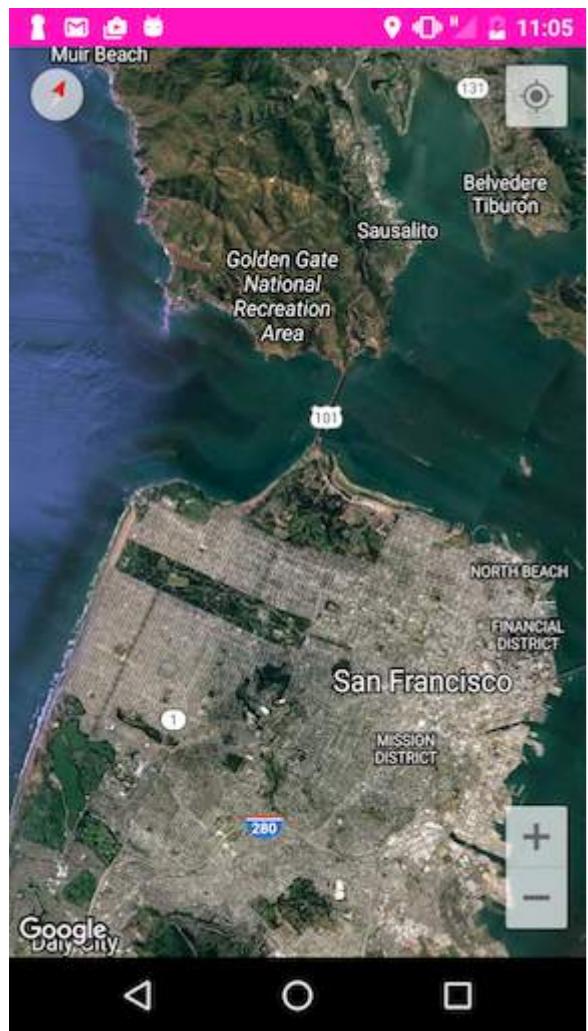
## Section 239.8: UISettings

Using [UISettings](#), the appearance of the Google Map can be modified.

Here is an example of some common settings:

```
mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
mGoogleMap.getUiSettings().setMapToolbarEnabled(true);
mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
mGoogleMap.getUiSettings().setCompassEnabled(true);
```

Result:



## 第239.9节：信息窗口点击监听器

下面是如何为每个标记的 InfoWindow 点击事件定义不同操作的示例。

使用一个 HashMap，其中标记 ID 作为键，值为点击 InfoWindow 时应执行的对应操作。

然后，使用 OnInfoWindowClickListener 来处理用户点击 InfoWindow 的事件，并使用 HashMap 来确定应执行的操作。

在这个简单的示例中，我们将根据点击的是哪个标记的 InfoWindow 来打开不同的 Activity。

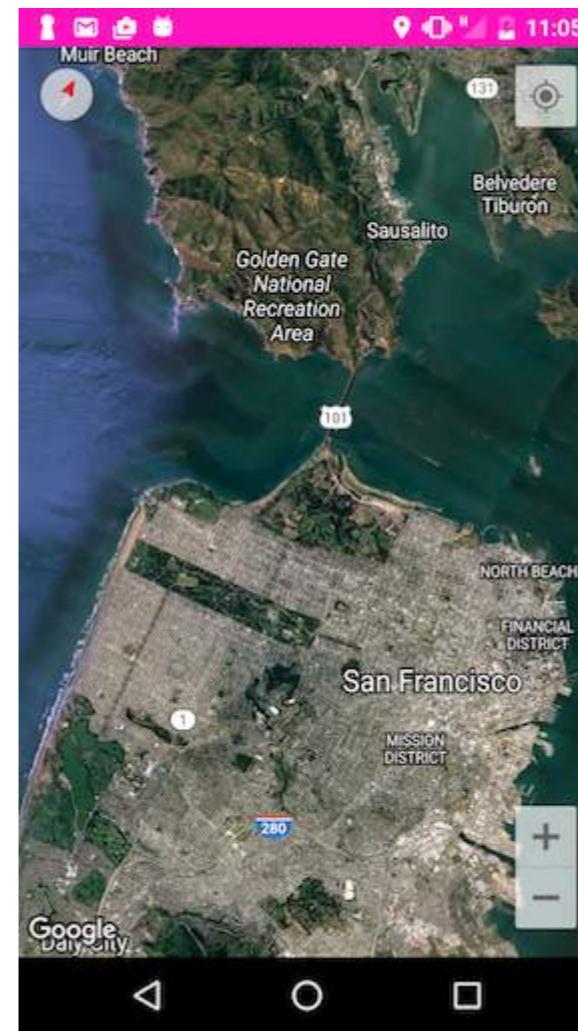
将 HashMap 声明为 Activity 或 Fragment 的实例变量：

```
//声明 HashMap 用于存储标记到 Activity 的映射关系
HashMap<String, String> markerMap = new HashMap<String, String>();
```

然后，每次添加标记时，在 HashMap 中添加一条条目，键为标记 ID，值为点击其 InfoWindow 时应执行的操作。

例如，添加两个标记并为每个标记定义一个操作：

```
Marker markerOne = googleMap.addMarker(new MarkerOptions().position(latLng1)
 .title("Marker One")
 .snippet("这是标记一"));
String idOne = markerOne.getId();
markerMap.put(idOne, "action_one");
```



## Section 239.9: InfoWindow Click Listener

Here is an example of how to define a different action for each Marker's InfoWindow click event.

Use a HashMap in which the marker ID is the key, and the value is the corresponding action it should take when the InfoWindow is clicked.

Then, use a OnInfoWindowClickListener to handle the event of a user clicking the InfoWindow, and use the HashMap to determine which action to take.

In this simple example we will open up a different Activity based on which Marker's InfoWindow was clicked.

Declare the HashMap as an instance variable of the Activity or Fragment:

```
//Declare HashMap to store mapping of marker to Activity
HashMap<String, String> markerMap = new HashMap<String, String>();
```

Then, each time you add a Marker, make an entry in the HashMap with the Marker ID and the action it should take when its InfoWindow is clicked.

For example, adding two Markers and defining an action to take for each:

```
Marker markerOne = googleMap.addMarker(new MarkerOptions().position(latLng1)
 .title("Marker One")
 .snippet("This is Marker One"));
String idOne = markerOne.getId();
markerMap.put(idOne, "action_one");
```

```

Marker markerTwo = googleMap.addMarker(new MarkerOptions().position(latLang2)
 .title("标记二")
 .snippet("这是标记二");
String idTwo = markerTwo.getId();
markerMap.put(idTwo, "action_two");

```

在信息窗口点击监听器中，从HashMap获取动作，并根据标记的动作打开相应的活动：

```

mGoogleMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
 @Override
 public void onInfoWindowClick(Marker marker) {
 String actionId = markerMap.get(marker.getId());
 if (actionId.equals("action_one")) {
 Intent i = new Intent(MainActivity.this, ActivityOne.class);
 startActivity(i);
 } 否则如果 (actionId.等于("action_two")) {
 Intent i = new Intent(MainActivity.this, ActivityTwo.class);
 startActivity(i);
 }
 }
});

```

注意 如果代码在 Fragment 中，需将 MainActivity.this 替换为 getActivity()。

## 第239.10节：获取证书密钥库文件的 SH1 指纹

为了为您的证书获取 Google 地图 API 密钥，您必须向 API 控制台提供调试/发布密钥库的 SH1 指纹。

您可以使用 JDK 的 keytool 程序获取密钥库，具体方法请参见 [此处](#) 的文档。

另一种方法是通过程序运行此代码片段，使用带有调试/发布证书签名的应用，并将哈希打印到日志中，以编程方式获取指纹。

```

PackageManager info;
尝试 {
 info = getPackageManager().getPackageInfo("com.package.name", PackageManager.GET_SIGNATURES);
 对于 (Signature signature : info.signatures) {
 MessageDigest md;
 md = MessageDigest.getInstance("SHA");
 md.update(signature.toByteArray());
 String hash= new String(Base64.encode(md.digest(), 0));
 Log.e("hash", hash);
 }
} 捕获 (NameNotFoundException e1) {
 Log.e("未找到名称", e1.toString());
} 捕获 (NoSuchAlgorithmException e) {
 Log.e("无此算法", e.toString());
} 捕获 (Exception e) {
 Log.e("异常", e.toString());
}

```

```

Marker markerTwo = googleMap.addMarker(new MarkerOptions().position(latLang2)
 .title("Marker Two")
 .snippet("This is Marker Two");
String idTwo = markerTwo.getId();
markerMap.put(idTwo, "action_two");

```

In the InfoWindow click listener, get the action from the HashMap, and open up the corresponding Activity based on the action of the Marker:

```

mGoogleMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
 @Override
 public void onInfoWindowClick(Marker marker) {
 String actionId = markerMap.get(marker.getId());
 if (actionId.equals("action_one")) {
 Intent i = new Intent(MainActivity.this, ActivityOne.class);
 startActivity(i);
 } else if (actionId.equals("action_two")) {
 Intent i = new Intent(MainActivity.this, ActivityTwo.class);
 startActivity(i);
 }
 }
});

```

Note If the code is in a Fragment, replace MainActivity.this with getActivity().

## Section 239.10: Obtaining the SH1-Fingerprint of your certificate keystore file

In order to obtain a Google Maps API key for your certificate, you must provide the API console with the SH1-fingerprint of your debug/release keystore.

You can obtain the keystore by using the **JDK's keytool** program as described [here](#) in the docs.

Another approach is to obtain the fingerprint programmatically by running this snippet with your app signed with the debug/release certificate and printing the hash to the log.

```

PackageManager info;
try {
 info = getPackageManager().getPackageInfo("com.package.name", PackageManager.GET_SIGNATURES);
 for (Signature signature : info.signatures) {
 MessageDigest md;
 md = MessageDigest.getInstance("SHA");
 md.update(signature.toByteArray());
 String hash= new String(Base64.encode(md.digest(), 0));
 Log.e("hash", hash);
 }
} catch (NameNotFoundException e1) {
 Log.e("name not found", e1.toString());
} catch (NoSuchAlgorithmException e) {
 Log.e("no such an algorithm", e.toString());
} catch (Exception e) {
 Log.e("exception", e.toString());
}

```

## 第239.11节：点击地图时不要启动谷歌地图（简化模式）

当谷歌地图以简化模式显示时，点击地图会打开谷歌地图应用。要禁用此功能，必须在MapView上调用setClickable(false)，例如：

```
final MapView mapView = (MapView) view.findViewById(R.id.map);
mapView.setClickable(false);
```

## Section 239.11: Do not launch Google Maps when the map is clicked (lite mode)

When a Google Map is displayed in lite mode clicking on a map will open the Google Maps application. To disable this functionality you must call `setClickable(false)` on the MapView, e.g.:

```
final MapView mapView = (MapView) view.findViewById(R.id.map);
mapView.setClickable(false);
```

# 第240章：谷歌云端硬盘API

谷歌云端硬盘是由谷歌创建的文件托管服务。它提供文件存储服务，允许用户将文件上传到云端，并与他人共享。通过使用谷歌云端硬盘API，我们可以实现计算机或移动设备与谷歌云端硬盘云之间的文件同步。

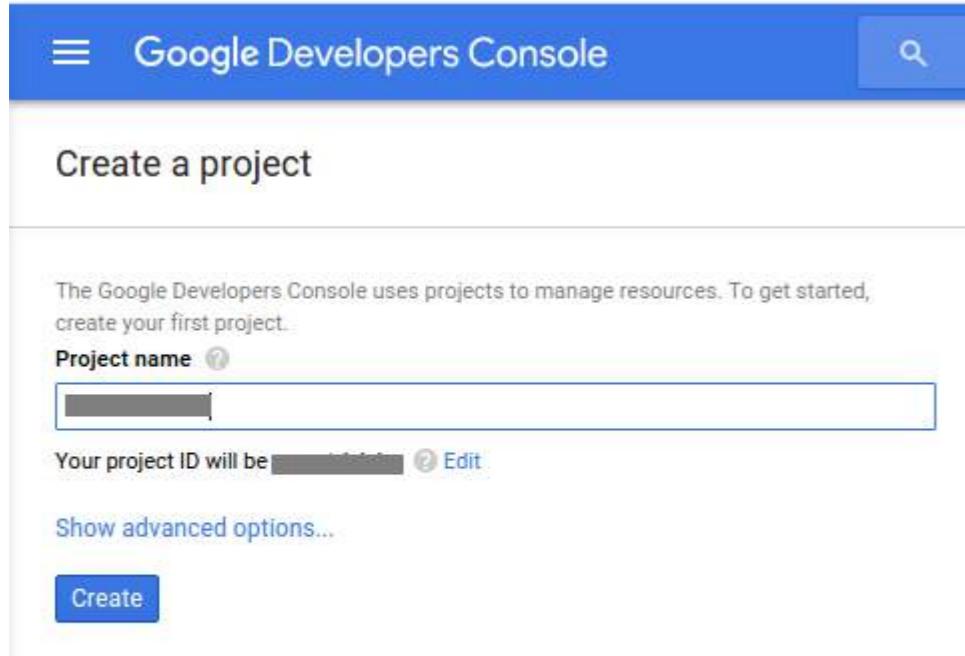
## 第240.1节：在安卓中集成谷歌云端硬盘

### 在谷歌开发者控制台创建新项目

为了将安卓应用与谷歌云端硬盘集成，需要在谷歌开发者控制台创建项目的凭据。因此，我们需要在谷歌开发者控制台创建一个项目。

在谷歌开发者控制台创建项目，请按照以下步骤操作：

- 访问[谷歌开发者控制台](#)的安卓部分。在输入框中填写你的**项目名称**，然后点击**创建**按钮，在谷歌开发者控制台创建一个新项目。



- 我们需要创建凭据以访问API。因此，点击创建凭据按钮。

# Chapter 240: Google Drive API

Google Drive is a file hosting service created by **Google**. It provides file storage service and allows the user to upload files in the cloud and also share with other people. Using Google Drive API, we can synchronize files between computer or mobile device and Google Drive Cloud.

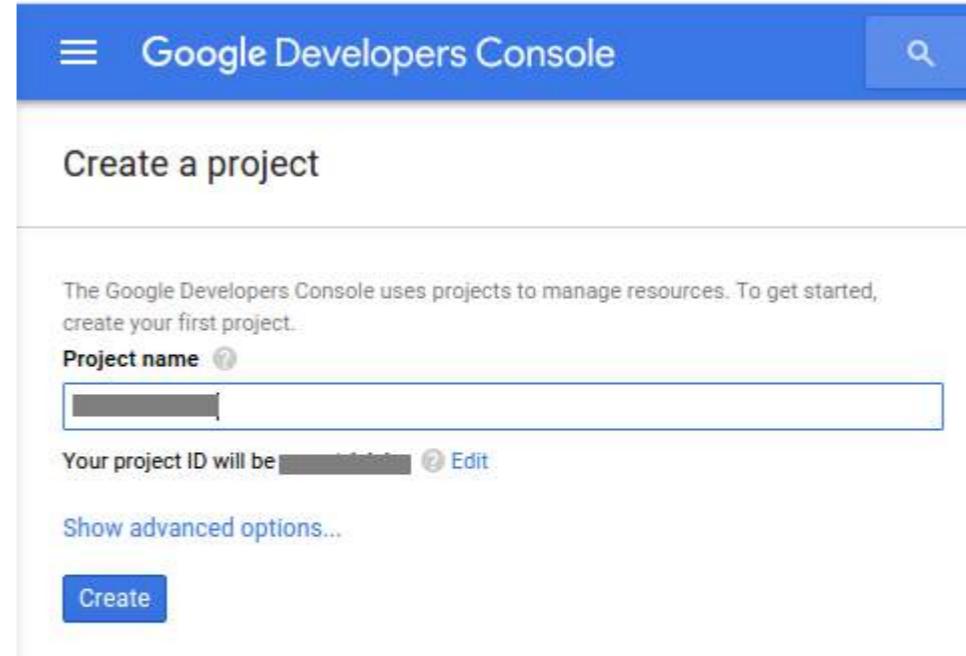
## Section 240.1: Integrate Google Drive in Android

### Create a New Project on Google Developer Console

To integrate Android application with Google Drive, create the credentials of project in the Google Developers Console. So, we need to create a project on Google Developer console.

To create a project on Google Developer Console, follow these steps:

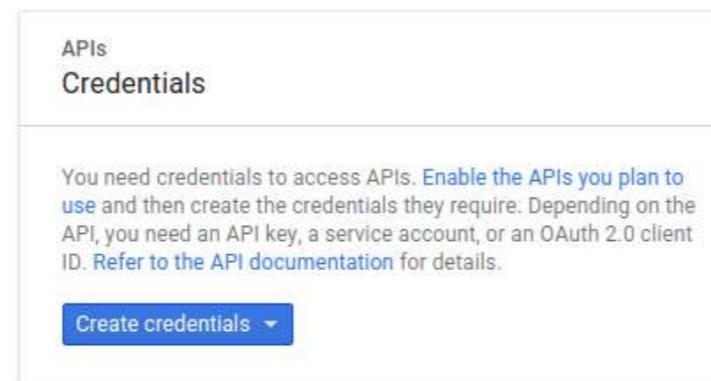
- Go to [Google Developer Console](#) for Android. Fill your **project name** in the input field and click on the **create** button to create a new project on Google Developer console.



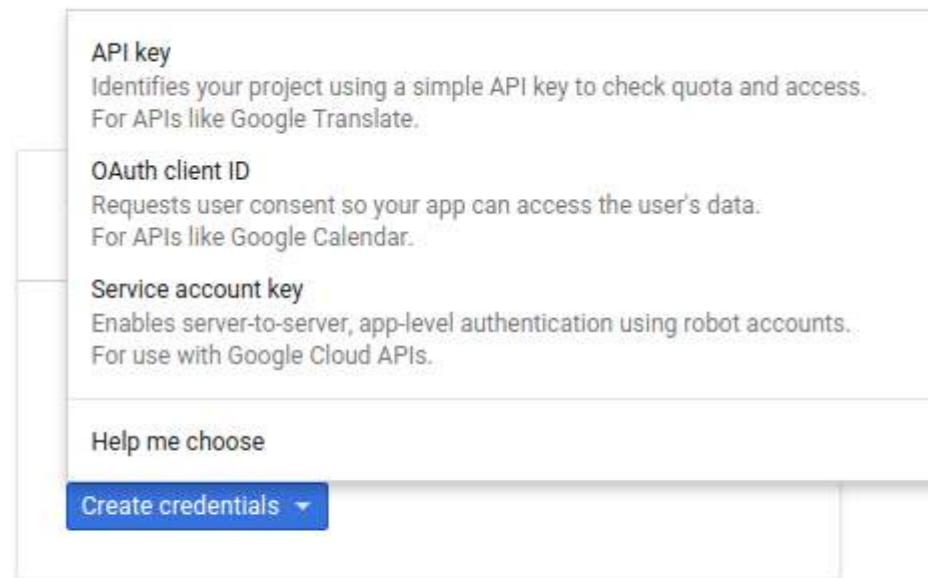
- We need to create credentials to access API. So, click on the **Create credentials** button.

## Credentials

Credentials OAuth consent screen Domain verification



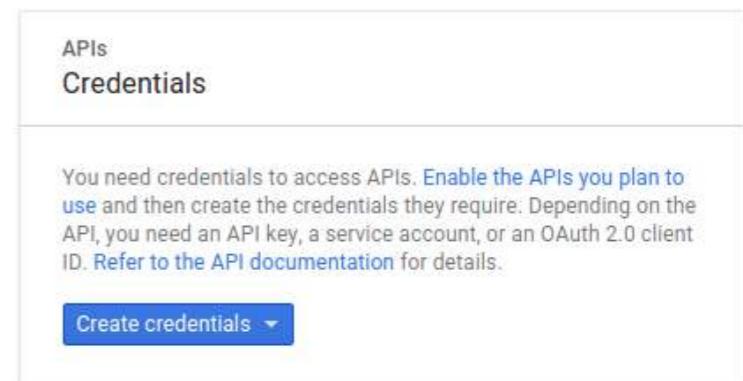
- 现在会弹出一个窗口。在列表中点击API密钥选项以创建API密钥。



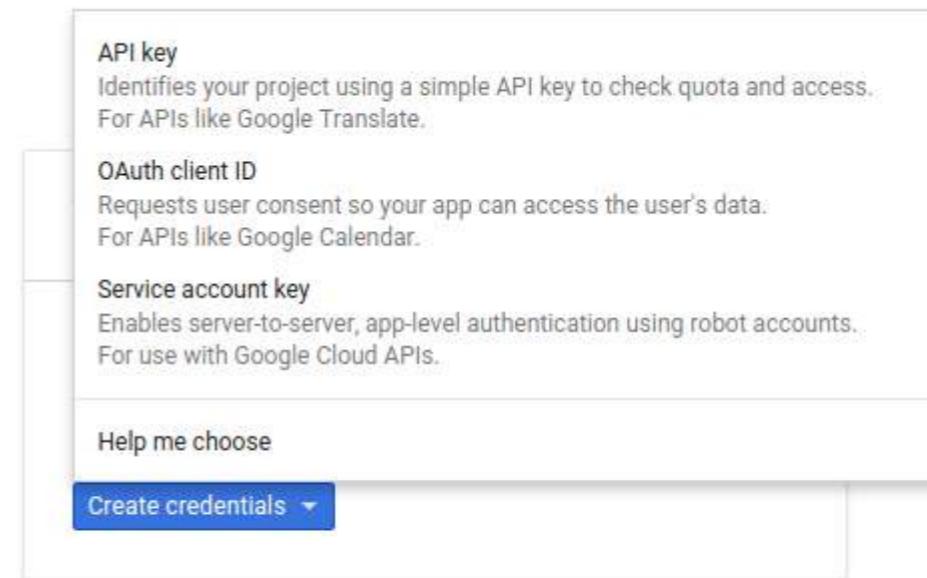
- 我们需要API密钥来调用安卓的谷歌API。因此，点击**安卓密钥**以识别你的安卓项目。

## Credentials

Credentials OAuth consent screen Domain verification



- Now, a pop window will open. Click on **API Key** option in the list to create API key.



- We need an API key to call Google APIs for Android. So, click on the **Android Key** to identify your Android Project.

The screenshot shows the Google Developers Console API Manager Credentials page. The left sidebar has 'API Manager' selected under 'API'. The main area is titled 'Create a new key' with the sub-instruction: 'You need an API key to call certain Google APIs. The API key identifies your project. Also, it is used to enforce quotas and handle billing, so keep it safe.' Below this are four tabs: 'Server key', 'Browser key', 'Android key', and 'iOS key'. The 'Android key' tab is currently selected.

- 接下来，我们需要在输入框中添加安卓项目的包名和**SHA-1指纹**以创建API密钥。

The screenshot shows the 'Create Android API key' page. The left sidebar has 'API Manager' selected under 'API'. The main area has a 'Name' input field containing 'Android key 1'. Below it is a note: 'Restrict usage to your Android apps (Optional)' with a detailed explanation. A command line instruction 'keytool -list -v -keystore mystore.keystore' is shown. There's a table for adding package names and fingerprints, with one entry: 'com.example' and '12:34:56:78:90:AB:CD:EF:12:34:56:78:90:AB:CD:EF:AA:BB:CC:DD'. A button '+ Add package name and fingerprint' is available. A note at the bottom says 'Note: It may take up to 5 minutes for settings to take effect'. At the bottom are 'Create' and 'Cancel' buttons.

- 我们需要生成**SHA-1指纹**。因此，打开终端并运行**Keytool工具**以获取SHA1指纹。在运行Keytool工具时，需要提供**密钥库密码**。默认开发keytool密码是“**android**”。  

```
keytool -exportcert -alias androiddebugkey -keystore
~/android/debug.keystore -list -v
```

The screenshot shows the 'Create a new key' page again, identical to the first one but with a different URL. The left sidebar has 'API Manager' selected under 'API'. The main area is titled 'Create a new key' with the same instructions and tabs: 'Server key', 'Browser key', 'Android key', and 'iOS key'. The 'Android key' tab is selected.

- Next, we need to add Package Name of the Android Project and **SHA-1 fingerprint** in the input fields to create API key.

The screenshot shows the 'Create a new key' page again, identical to the first one but with a different URL. The left sidebar has 'API Manager' selected under 'API'. The main area is titled 'Create a new key' with the same instructions and tabs: 'Server key', 'Browser key', 'Android key', and 'iOS key'. The 'Android key' tab is selected.

- We need to generate **SHA-1 fingerprint**. So, open your terminal and run **Keytool utility** to get the SHA1 fingerprint. While running Keytool utility, you need to provide **keystore password**. Default development keytool password is “**android**”.  

```
keytool -exportcert -alias androiddebugkey -keystore
~/android/debug.keystore -list -v
```

```

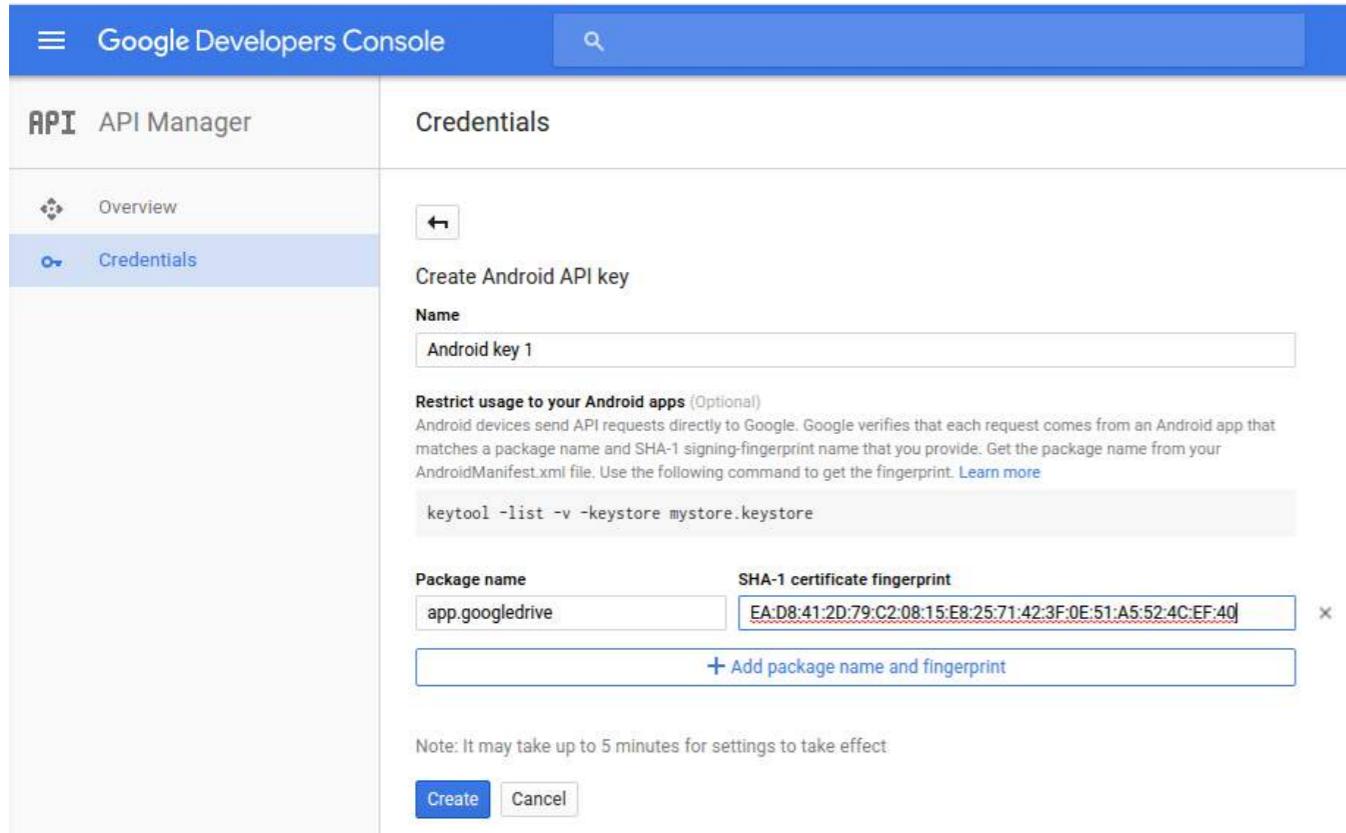
[...@...: ~]$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
MD5: 77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C4:F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73 6B 67 99 5A A3 C0 05 4A skg.Z...
0010: 93 BE 25 4F ..%0
]
]

```

- 现在，在凭据页面的输入框中添加**包名**和**SHA-1指纹**。最后，点击创建按钮以创建API密钥。



- 这将为Android创建API密钥。我们将使用此API密钥将Android应用与Google Drive集成。

```

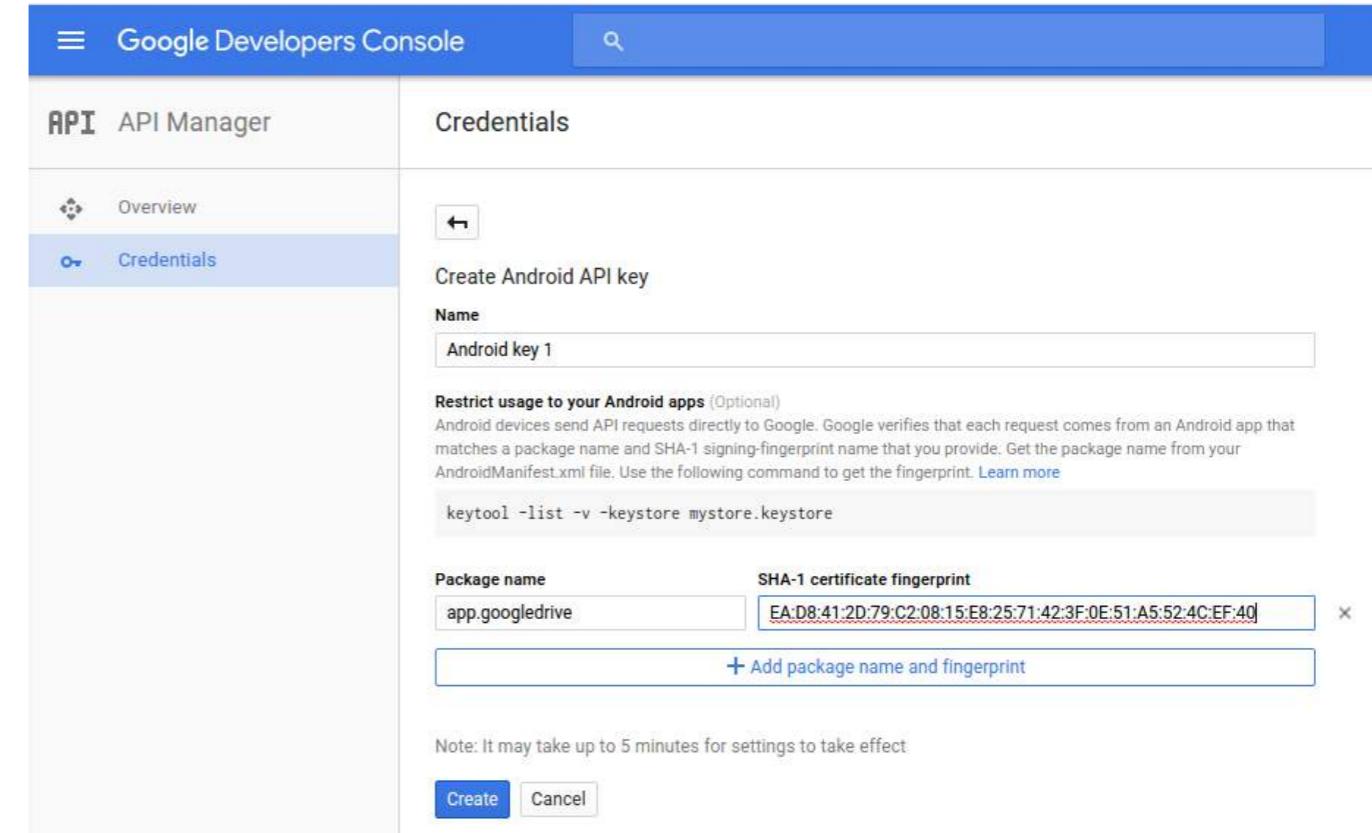
[...@...: ~]$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
MD5: 77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C4:F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73 6B 67 99 5A A3 C0 05 4A skg.Z...
0010: 93 BE 25 4F ..%0
]
]

```

- Now, add **Package name** and **SHA-1 fingerprint** in input fields on credentials page. Finally, click on create button to create API key.



- This will create API key for Android. We will use the this API key to integrate Android application with Google Drive.

## Credentials

Credentials OAuth consent screen Domain verification

Create credentials  Delete

Create credentials to access your enabled APIs. Refer to the API documentation for details.

### API keys

| <input type="checkbox"/> Name          | Creation date | Type    | Key                                |
|----------------------------------------|---------------|---------|------------------------------------|
| <input type="checkbox"/> Android key 1 | Mar 9, 2016   | Android | XlzaSyAr_XXXXXXXXXXXXXXXXXXXX-XXXX |

## Credentials

Credentials OAuth consent screen Domain verification

Create credentials  Delete

Create credentials to access your enabled APIs. Refer to the API documentation for details.

### API keys

| <input type="checkbox"/> Name          | Creation date | Type    | Key                                |
|----------------------------------------|---------------|---------|------------------------------------|
| <input type="checkbox"/> Android key 1 | Mar 9, 2016   | Android | XlzaSyAr_XXXXXXXXXXXXXXXXXXXX-XXXX |

## 启用Google Drive API

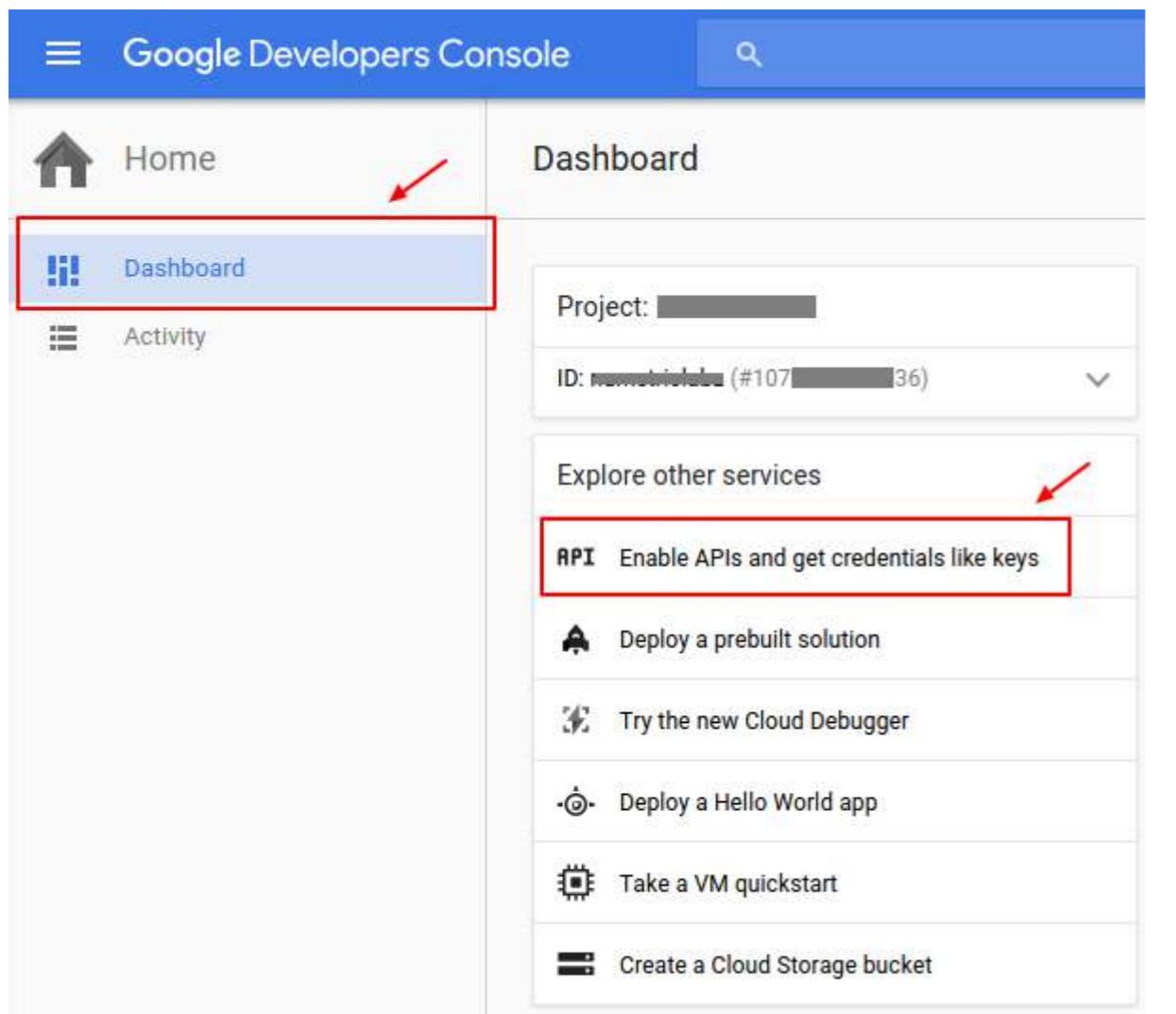
我们需要启用Google Drive API，以便从Android应用访问存储在Google Drive上的文件。要启用Google Drive API，请按照以下步骤操作：

- 进入你的[Google开发者控制台仪表板](#)，点击[启用API并获取凭据（如密钥）](#)，然后你将看到流行的Google API列表。

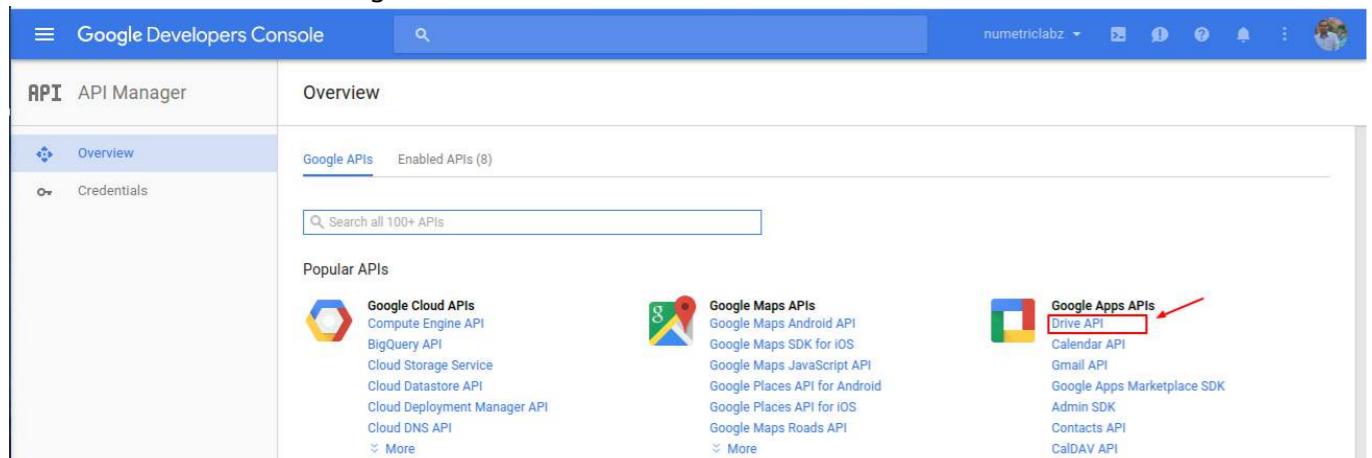
## Enable Google Drive API

We need to enable Google Drive API to access files stored on Google Drive from Android application. To enable Google Drive API, follow below steps:

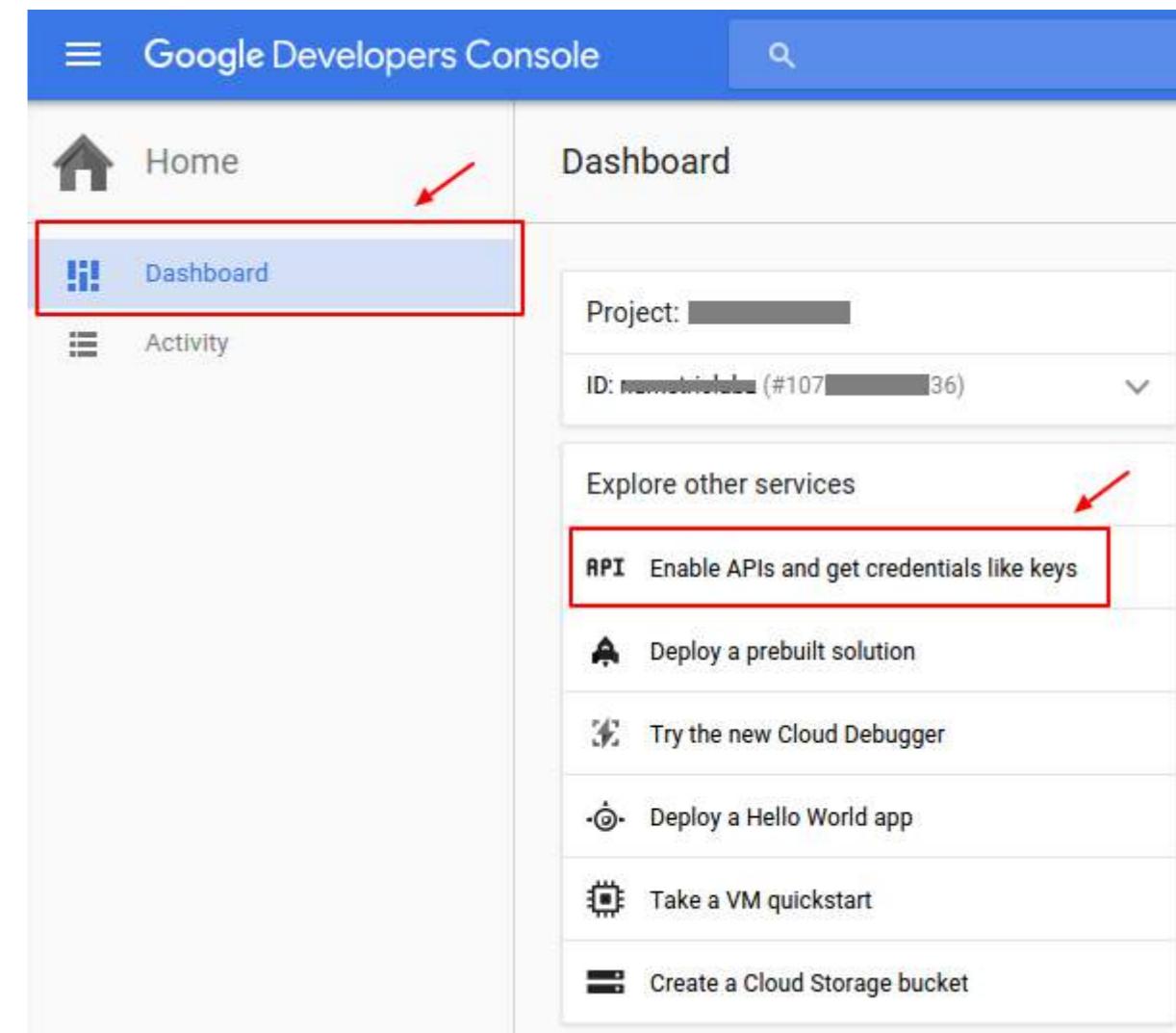
- Go to your [Google Developer console Dashboard](#) and click on **Enable APIs get credentials like keys** then you will see popular Google APIs.



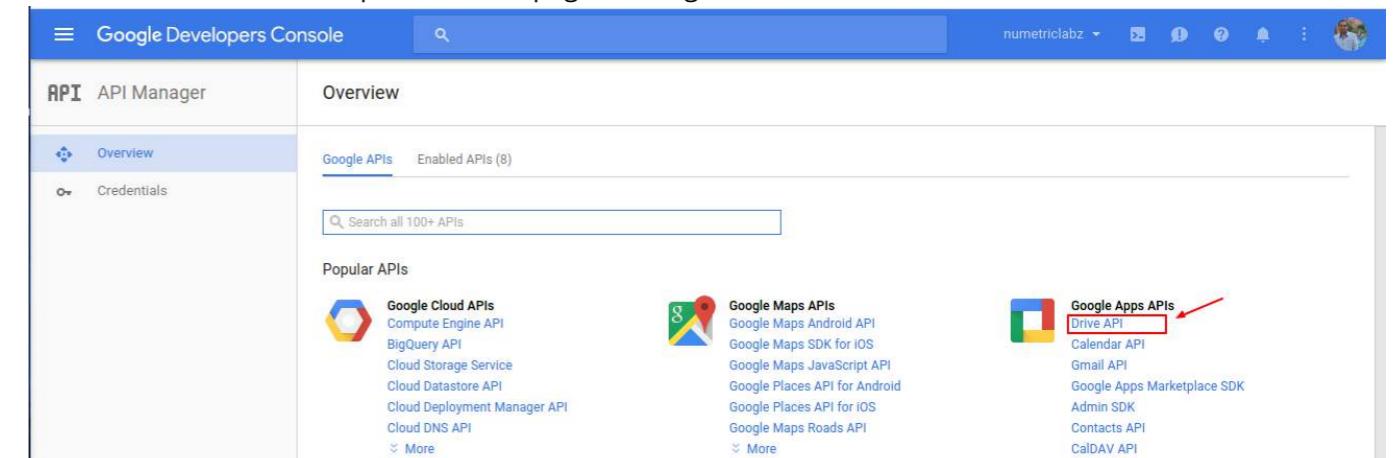
- 点击Drive API链接以打开Google Drive API的概览页面。



- 点击启用按钮以启用Google云端硬盘API。它允许客户端访问Google云端硬盘。



- Click on **Drive API** link to open overview page of Google Drive API.



- Click on the Enable button to enable Google drive API. It allows client access to Google Drive.

The screenshot shows the Google Developers Console API Manager Overview page for the Drive API. The left sidebar has 'API Manager' selected under 'API'. The main area shows the 'Drive API' with a status of 'Enabled'. It includes a brief description: 'The Drive API allows clients to access resources from Google Drive.', a 'Learn more' link, and a 'Try this API in APIs Explorer' button.

This screenshot is identical to the one above, showing the Google Developers Console API Manager Overview page for the Drive API. The Drive API is enabled, and its details are displayed in the main area.

## 添加网络权限

应用需要网络访问Google云端硬盘文件。使用以下代码在

AndroidManifest.xml文件中设置网络权限：

```
<uses-permission android:name="android.permission.INTERNET" />
```

## 添加Google Play服务

我们将使用**Google Play服务API** 其中包括**Google云端硬盘Android API** 因此，我们需要在Android应用中设置Google Play服务SDK。打开你的build.gradle（应用模块）文件，并将Google Play服务SDK添加为依赖项。

```
dependencies {
 ...
 compile 'com.google.android.gms:play-services:<latest_version>'
 ...
}
```

## 在Manifest文件中添加API密钥

要在安卓应用中使用谷歌API，我们需要在AndroidManifest.xml文件中添加API密钥和Google Play服务的版本。在AndroidManifest.xml文件的标签内添加正确的元数据标签。

## 连接并授权谷歌云端硬盘安卓API

我们需要对谷歌云端硬盘安卓API进行身份验证并与安卓应用连接。谷歌云端硬盘安卓API的授权由GoogleApiClient处理。我们将在onResume()方法中使用GoogleApiClient。

```
/**
 * 当活动开始与用户交互时调用。
 * 此时您的活动处于活动栈的顶部,

```

## Add Internet Permission

App needs **Internet** access Google Drive files. Use the following code to set up Internet permissions in AndroidManifest.xml file :

```
<uses-permission android:name="android.permission.INTERNET" />
```

## Add Google Play Services

We will use **Google play services API** which includes the **Google Drive Android API**. So, we need to setup Google play services SDK in Android Application. Open your build.gradle(app module) file and add Google play services SDK as a dependencies.

```
dependencies {
 ...
 compile 'com.google.android.gms:play-services:<latest_version>'
 ...
}
```

## Add API key in Manifest file

To use Google API in Android application, we need to add API key and version of the Google Play Service in the AndroidManifest.xml file. Add the correct metadata tags inside the tag of the AndroidManifest.xml file.

## Connect and Authorize the Google Drive Android API

We need to authenticate and connect **Google Drive Android API** with Android application. Authorization of **Google Drive Android API** is handled by the **GoogleApiClient**. We will use **GoogleApiClient** within **onResume()** method.

```
/**
 * Called when the activity will start interacting with the user.
 * At this point your activity is at the top of the activity stack,
```

```

* 用户输入将传递给该活动。
*/
@Override
protected void onResume() {
 super.onResume();
 if (mGoogleApiClient == null) {

 /**
 * 创建API客户端并绑定到实例变量。
 * 我们使用此实例作为连接和连接失败的回调。
 * 由于未传递账户名称，系统会提示用户进行选择。
 */
 mGoogleApiClient = new GoogleApiClient.Builder(this)
 .addApi(Drive.API)
 .addScope(Drive.SCOPE_FILE)
 .addConnectionCallbacks(this)
 .addOnConnectionFailedListener(this)
 .build();
 }

 mGoogleApiClient.connect();
}

```

## 断开 Google Drive Android API 连接

当活动停止时，我们将在活动的 `onStop()` 方法中调用 `disconnect()` 方法，断开 Android 应用与 Google Drive Android API 的连接。

```

@Override
protected void onStop() {
 super.onStop();
 if (mGoogleApiClient != null) {

 // 断开 Google Android Drive API 连接。
 mGoogleApiClient.disconnect();
 }
 super.onPause();
}

```

## 实现连接回调和连接失败监听器

我们将在 `MainActivity.java` 文件中实现 Google API 客户端的连接回调（`Connection Callbacks`）和连接失败监听器（`Connection Failed Listener`），以了解 Google API 客户端的连接状态。这些监听器提供了 `onConnected()`、`onConnectionFailed()`、`onConnectionSuspended()` 方法，用于处理应用与 Drive 之间的连接问题。

如果用户已授权该应用，则会调用 `onConnected()` 方法。如果用户未授权应用，则会调用 `onConnectionFailed()` 方法，并向用户显示一个对话框，提示您的应用未被授权访问 Google Drive。如果连接被中断，则会调用 `onConnectionSuspended()` 方法。

您需要在活动中实现 `ConnectionCallbacks` 和 `OnConnectionFailedListener`。请在您的 Java 文件中使用以下代码。

```

@Override
public void onConnectionFailed(ConnectionResult result) {

 // 当 API 客户端连接失败时调用。
 Log.i(TAG, "GoogleApiClient 连接失败：" + result.toString());
}

```

```

* with user input going to it.
*/
@Override
protected void onResume() {
 super.onResume();
 if (mGoogleApiClient == null) {

 /**
 * Create the API client and bind it to an instance variable.
 * We use this instance as the callback for connection and connection failures.
 * Since no account name is passed, the user is prompted to choose.
 */
 mGoogleApiClient = new GoogleApiClient.Builder(this)
 .addApi(Drive.API)
 .addScope(Drive.SCOPE_FILE)
 .addConnectionCallbacks(this)
 .addOnConnectionFailedListener(this)
 .build();
 }

 mGoogleApiClient.connect();
}

```

## Disconnect Google Deive Android API

When activity stops, we will disconnected Google Drive Android API connection with Android application by calling `disconnect()` method inside activity's `onStop()` method.

```

@Override
protected void onStop() {
 super.onStop();
 if (mGoogleApiClient != null) {

 // disconnect Google Android Drive API connection.
 mGoogleApiClient.disconnect();
 }
 super.onPause();
}

```

## Implement Connection Callbacks and Connection Failed Listener

We will implement Connection Callbacks and Connection Failed Listener of Google API client in `MainActivity.java` file to know status about connection of Google API client. These listeners provide `onConnected()`, `onConnectionFailed()`, `onConnectionSuspended()` method to handle the connection issues between app and Drive.

If user has authorized the application, the `onConnected()` method is invoked. If user has not authorized application, `onConnectionFailed()` method is invoked and a dialog is displayed to user that your app is not authorized to access Google Drive. In case connection is suspended, `onConnectionSuspended()` method is called.

You need to implement `ConnectionCallbacks` and `OnConnectionFailedListener` in your activity. Use the following code in your Java file.

```

@Override
public void onConnectionFailed(ConnectionResult result) {

 // Called whenever the API client fails to connect.
 Log.i(TAG, "GoogleApiClient connection failed:" + result.toString());
}

```

```

if (!result.hasResolution()) {

 // 显示本地化的错误对话框。
 GoogleApiAvailability.getInstance().getErrorDialog(this, result.getErrorCode(),
 0).show();
 return;
}

/**
* 该故障有解决方案。请解决它。
* 通常在应用尚未授权时调用，并向用户显示授权对话框。
*/
try {

 result.startResolutionForResult(this, REQUEST_CODE_RESOLUTION);

} catch (SendIntentException e) {

 Log.e(TAG, "启动解决活动时异常", e);
}

/**
* 当Google API客户端连接时调用
* @param connectionHint
*/
@Override
public void onConnected(Bundle connectionHint) {

 Toast.makeText(getApplicationContext(), "已连接", Toast.LENGTH_LONG).show();
}

/**
* 当连接中断时调用
* @param cause
*/
@Override
public void onConnectionSuspended(int cause) {

 Log.i(TAG, "GoogleApiClient 连接已暂停");
}

```

## 第240.2节：在Google云端硬盘上创建文件

我们将在Google云端硬盘上添加一个文件。我们将使用Drive对象的createFile()方法在Google云端硬盘上以编程方式创建文件。在此示例中，我们将在用户的根文件夹中添加一个新的文本文件。当添加文件时，需要指定初始的元数据集合、文件内容和父文件夹。

我们需要创建一个CreateMyFile()回调方法，在该方法中，使用Drive对象获取一个DriveContents资源。然后将API客户端传递给Drive对象，并调用driveContentsCallback回调方法来处理DriveContents的结果。

一个DriveContents资源包含文件二进制流的临时副本，该副本仅对应用程序可用。

```

public void CreateMyFile(){
 fileOperation = true;
}

```

```

if (!result.hasResolution()) {

 // show the localized error dialog.
 GoogleApiAvailability.getInstance().getErrorDialog(this, result.getErrorCode(),
 0).show();
 return;
}

/**
* The failure has a resolution. Resolve it.
* Called typically when the app is not yet authorized, and an authorization
* dialog is displayed to the user.
*/
try {

 result.startResolutionForResult(this, REQUEST_CODE_RESOLUTION);

} catch (SendIntentException e) {

 Log.e(TAG, "Exception while starting resolution activity", e);
}

/**
* It invoked when Google API client connected
* @param connectionHint
*/
@Override
public void onConnected(Bundle connectionHint) {

 Toast.makeText(getApplicationContext(), "Connected", Toast.LENGTH_LONG).show();
}

/**
* It invoked when connection suspended
* @param cause
*/
@Override
public void onConnectionSuspended(int cause) {

 Log.i(TAG, "GoogleApiClient connection suspended");
}

```

## Section 240.2: Create a File on Google Drive

We will add a file on Google Drive. We will use the `createFile()` method of a `Drive` object to create file programmatically on Google Drive. In this example we are adding a new text file in the user's root folder. When a file is added, we need to specify the initial set of metadata, file contents, and the parent folder.

We need to create a `CreateMyFile()` callback method and within this method, use the `Drive` object to retrieve a `DriveContents` resource. Then we pass the API client to the `Drive` object and call the `driveContentsCallback` callback method to handle result of `DriveContents`.

A `DriveContents` resource contains a temporary copy of the file's binary stream which is only available to the application.

```

public void CreateMyFile(){
 fileOperation = true;
}

```

```
// 创建新的内容资源。
Drive.DriveApi.newDriveContents(mGoogleApiClient)
 .setResultCallback(driveContentsCallback);
}
```

## DriveContents的结果处理器

处理响应需要检查调用是否成功。如果调用成功，我们可以获取DriveContents资源。

我们将创建一个DriveContents的结果处理器。在此方法中，我们调用CreateFileOnGoogleDrive()方法并传递DriveContentsResult的结果：

```
/**
 * 这是Drive内容的结果处理器。
 * 此回调方法调用CreateFileOnGoogleDrive()方法。
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
 new ResultCallback<DriveContentsResult>() {
 @Override
 public void onResult(DriveContentsResult result) {
 if (result.getStatus().isSuccess()) {
 if (fileOperation == true){
 CreateFileOnGoogleDrive(result);
 }
 }
 }
};
```

## 编程方式创建文件

要创建文件，我们需要使用一个MetadataChangeSet对象。通过使用该对象，我们设置标题（文件名）和文件类型。同时，我们必须使用DriveFolder类的createFile()方法，并传入Google客户端API、MetaDataSet对象和driveContents来创建文件。我们调用结果处理器回调来处理创建文件的结果。

我们使用以下代码在用户的根文件夹中创建一个新的文本文件：

```
/**
 * 使用 MetadataChangeSet 对象在根文件夹中创建文件。
 * @param result
 */
public void CreateFileOnGoogleDrive(DriveContentsResult result){

 final DriveContents driveContents = result.getDriveContents();

 // 在非UI线程执行I/O操作。
 new Thread() {
 @Override
 public void run() {
 // 向 DriveContents 写入内容。
 OutputStream outputStream = driveContents.getOutputStream();
 Writer writer = new OutputStreamWriter(outputStream);
 try {
writer.write("Hello Christlin!");
 writer.close();
 } catch (IOException e) {
 Log.e(TAG, e.getMessage());
 }
 }
 };
}

MetadataChangeSet changeSet = new MetadataChangeSet.Builder()
```

```
// Create new contents resource.
Drive.DriveApi.newDriveContents(mGoogleApiClient)
 .setResultCallback(driveContentsCallback);
}
```

## Result Handler of DriveContents

Handling the response requires to check if the call was successful or not. If the call was successful, we can retrieve the DriveContents resource.

We will create a result handler of DriveContents. Within this method, we call the CreateFileOnGoogleDrive() method and pass the result of DriveContentsResult:

```
/**
 * This is the Result result handler of Drive contents.
 * This callback method calls the CreateFileOnGoogleDrive() method.
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
 new ResultCallback<DriveContentsResult>() {
 @Override
 public void onResult(DriveContentsResult result) {
 if (result.getStatus().isSuccess()) {
 if (fileOperation == true){
 CreateFileOnGoogleDrive(result);
 }
 }
 }
};
```

## Create File Programmatically

To create files, we need to use a MetadataChangeSet object. By using this object, we set the title (file name) and file type. Also, we must use the createFile() method of the DriveFolder class and pass the Google client API, the MetaDataSet object, and the driveContents to create a file. We call the result handler callback to handle the result of the created file.

We use the following code to create a new text file in the user's root folder:

```
/**
 * Create a file in the root folder using a MetadataChangeSet object.
 * @param result
 */
public void CreateFileOnGoogleDrive(DriveContentsResult result){

 final DriveContents driveContents = result.getDriveContents();

 // Perform I/O off the UI thread.
 new Thread() {
 @Override
 public void run() {
 // Write content to DriveContents.
 OutputStream outputStream = driveContents.getOutputStream();
 Writer writer = new OutputStreamWriter(outputStream);
 try {
writer.write("Hello Christlin!");
 writer.close();
 } catch (IOException e) {
 Log.e(TAG, e.getMessage());
 }
 }
 };
}

MetadataChangeSet changeSet = new MetadataChangeSet.Builder()
```

```

.setTitle("我的第一个Drive文件")
.setMimeType("text/plain")
.setStarred(true).build();

// 在根文件夹中创建文件。
Drive.DriveApi.getRootFolder(mGoogleApiClient)
.createFile(mGoogleApiClient, changeSet, driveContents)
setResultCallback(fileCallback);
}
}.start();
}

```

## 处理已创建文件的结果

以下代码将创建一个回调方法来处理已创建文件的结果：

```

/**
* 处理已创建文件的结果
*/
final private ResultCallback<DriveFolder.DriveFileResult> fileCallback = new
 ResultCallback<DriveFolder.DriveFileResult>() {
 @Override
 public void onResult(DriveFolder.DriveFileResult result) {
 if (result.getStatus().isSuccess()) {
 Toast.makeText(getApplicationContext(), "文件已创建: "+
 result.getDriveFile().getDriveId(), Toast.LENGTH_LONG).show();
 }
 return;
 }
 };
}

```

```

.setTitle("My First Drive File")
.setMimeType("text/plain")
.setStarred(true).build();

// Create a file in the root folder.
Drive.DriveApi.getRootFolder(mGoogleApiClient)
.createFile(mGoogleApiClient, changeSet, driveContents)
setResultCallback(fileCallback);
}
}.start();
}

```

## Handle result of Created File

The following code will create a callback method to handle the result of the created file:

```

/**
* Handle result of Created file
*/
final private ResultCallback<DriveFolder.DriveFileResult> fileCallback = new
 ResultCallback<DriveFolder.DriveFileResult>() {
 @Override
 public void onResult(DriveFolder.DriveFileResult result) {
 if (result.getStatus().isSuccess()) {
 Toast.makeText(getApplicationContext(), "file created: "+
 result.getDriveFile().getDriveId(), Toast.LENGTH_LONG).show();
 }
 return;
 }
 };
}

```

# 第241章：展示谷歌广告

## 第241.1节：添加插页广告

插页广告是覆盖其宿主应用界面的全屏广告。它们通常在应用流程中的自然过渡点显示，例如活动之间或游戏关卡暂停期间。

确保您的Manifest文件中具有必要的权限：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

1. 登录您的AdMob账户。
2. 点击Monetize标签。
- 3.选择或创建应用并选择平台。
- 4.选择插页广告并填写广告单元名称。
5. 创建广告单元后，您可以在仪表板上看到广告单元ID。例如：ca-app-pub-000000000000/000000000
6. 添加依赖项

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

这个应该放在底部。

```
apply plugin: 'com.google.gms.google-services'
```

将您的广告单元ID添加到strings.xml文件中

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

向您的清单文件添加ConfigChanges和元数据：

```
<activity
 android:name="com.google.android.gms.ads.AdActivity"
 android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"
 android:theme="@android:style/Theme.Translucent" />
```

和

```
<meta-data
 android:name="com.google.android.gms.version"
 android:value="@integer/google_play_services_version" />
```

活动：

```
public class AdActivity extends AppCompatActivity {
 private String TAG = AdActivity.class.getSimpleName();
 InterstitialAd mInterstitialAd;
```

# Chapter 241: Displaying Google Ads

## Section 241.1: Adding Interstitial Ad

Interstitial ads are full-screen ads that cover the interface of their host app. They're typically displayed at natural transition points in the flow of an app, such as between activities or during the pause between levels in a game.

Make sure you have necessary permissions in your [Manifest](#) file:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

1. Go to your [AdMob](#) account.
2. Click on **Monetize** tab.
3. Select or Create the app and choose the platform.
4. Select Interstitial and give an ad unit name.
5. Once the ad unit is created, you can notice the Ad unit ID on the dashboard. For example: ca-app-pub-000000000000/000000000
6. Add dependencies

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

This one should be on the bottom.

```
apply plugin: 'com.google.gms.google-services'
```

Add your **Ad unit ID** to your strings.xml file

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Add ConfigChanges and meta-data to your manifest:

```
<activity
 android:name="com.google.android.gms.ads.AdActivity"
 android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"
 android:theme="@android:style/Theme.Translucent" />
```

和

```
<meta-data
 android:name="com.google.android.gms.version"
 android:value="@integer/google_play_services_version" />
```

Activity:

```
public class AdActivity extends AppCompatActivity {
 private String TAG = AdActivity.class.getSimpleName();
 InterstitialAd mInterstitialAd;
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_second);

 mInterstitialAd = new InterstitialAd(this);

 // 设置广告单元ID
 mInterstitialAd.setAdUnitId(getString(R.string.interstitial_full_screen));

 AdRequest adRequest = new AdRequest.Builder()
 .build();

 // 加载插页式广告
 mInterstitialAd.loadAd(adRequest);

 mInterstitialAd.setAdListener(new AdListener() {
 public void onAdLoaded() {
 showInterstitial();
 }
 });
}

private void showInterstitial() {
 if (mInterstitialAd.isLoaded()) {
 mInterstitialAd.show();
 }
}
}

```

此 AdActivity 现在将显示全屏广告。

## 第241.2节：基础广告设置

您需要将以下内容添加到您的依赖项中：

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

然后将此内容放入同一文件中。

```
apply plugin: 'com.google.gms.google-services'
```

接下来，您需要将相关信息添加到您的 strings.xml 中。

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

接下来，将广告视图放置在您想要的位置，并像其他视图一样进行样式设置。

```

<com.google.android.gms.ads.AdView
 android:id="@+id/adView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_alignParentBottom="true"
 ads:adSize="BANNER"
 ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_second);

 mInterstitialAd = new InterstitialAd(this);

 // set the ad unit ID
 mInterstitialAd.setAdUnitId(getString(R.string.interstitial_full_screen));

 AdRequest adRequest = new AdRequest.Builder()
 .build();

 // Load ads into Interstitial Ads
 mInterstitialAd.loadAd(adRequest);

 mInterstitialAd.setAdListener(new AdListener() {
 public void onAdLoaded() {
 showInterstitial();
 }
 });
}

private void showInterstitial() {
 if (mInterstitialAd.isLoaded()) {
 mInterstitialAd.show();
 }
}
}

```

This AdActivity will show a full screen ad now.

## Section 241.2: Basic Ad Setup

You'll need to add the following to your dependencies:

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

and then put this in the same file.

```
apply plugin: 'com.google.gms.google-services'
```

Next you'll need to add relevant information into your strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Next place an advview wherever you want it and style it just like any other view.

```

<com.google.android.gms.ads.AdView
 android:id="@+id/adView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_alignParentBottom="true"
 ads:adSize="BANNER"
 ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>

```

最后但同样重要的是，将此代码放入你的 `onCreate` 方法中。

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

如果你完全复制粘贴，现在应该能看到一个小的横幅广告。只需在需要的地方放置更多的 `AdView` 即可显示更多广告。

And last but not least, throw this in your `onCreate`.

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

If you copy-pasted exactly you should now have a small banner ad. Simply place more `AdViews` wherever you need them for more.

# 第242章 : AdMob

## 参数

ads:adUnitId="@string/main\_screen\_ad" [1]

### 详细信息

您的广告ID。从Admob网站获取您的ID。“虽然这不是必需的将广告单元ID值存储在资源文件中是一种良好的做法。随着您的应用程序的发展和广告发布需求的成熟，可能需要更改ID值。如果您将它们保存在资源文件中，您就不必在代码中到处查找它们。”。[1]

## 第242.1节 : 实现

注意：此示例需要有效的Admob账户和有效的Admob广告代码。

### 应用级别的Build.gradle

如果已有版本，请更改为最新版本：

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

### 清单文件

访问广告数据需要互联网权限。请注意，此权限不必请求（使用API 23+），因为它是普通权限而非危险权限：

```
<uses-permission android:name="android.permission.INTERNET" />
```

### XML

以下XML示例展示了一个横幅广告：

```
<com.google.android.gms.ads.AdView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/adView"
 ads:adSize="BANNER"
 ads:adUnitId="@string/main_screen_ad" />
```

有关其他类型代码，请参阅Google AdMob帮助。

### Java

以下代码用于集成横幅广告。请注意，其他广告类型可能需要不同的集成方式：

```
// 更快初始化的替代方案。
// MobileAds.initialize(getApplicationContext(), "AD_UNIT_ID");

AdView mAdView = (AdView) findViewById(R.id.adView);
// 如果您在发布前进行测试，请添加您的设备测试ID。
// 设备测试ID可以在admob堆栈跟踪中找到。
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

在您的活动的 onResume()、onPause() 和 onDestroy() 方法中添加 AdView 生命周期方法：

```
@Override
public void onPause() {
 if (mAdView != null) {
 mAdView.pause();
```

# Chapter 242: AdMob

## Param

ads:adUnitId="@string/main\_screen\_ad" [1]

### Details

The ID of your ad. Get your ID from the admob site. "While it's not a requirement, storing your ad unit ID values in a resource file is a good practice. As your app grows and your ad publishing needs mature, it may be necessary to change the ID values. If you keep them in a resource file, you never have to search through your code looking for them.". [1]

## Section 242.1: Implementing

Note: This example requires a valid Admob account and valid Admob ad code.

### Build.gradle on app level

Change to the latest version if existing:

```
compile 'com.google.firebaseio:firebase-ads:10.2.1'
```

### Manifest

Internet permission is required to access the ad data. Note that this permission does not have to be requested (using API 23+) as it is a normal permission and not dangerous:

```
<uses-permission android:name="android.permission.INTERNET" />
```

### XML

The following XML example shows a banner ad:

```
<com.google.android.gms.ads.AdView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/adView"
 ads:adSize="BANNER"
 ads:adUnitId="@string/main_screen_ad" />
```

For the code of other types, refer to the [Google AdMob Help](#).

### Java

The following code is for the integration of banner ads. Note that other ad types may require different integration:

```
// Alternative for faster initialization.
// MobileAds.initialize(getApplicationContext(), "AD_UNIT_ID");

AdView mAdView = (AdView) findViewById(R.id.adView);
// Add your device test ID if you are doing testing before releasing.
// The device test ID can be found in the admob stacktrace.
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

Add the AdView life cycle methods in the onResume(), onPause(), and onDestroy() methods of your activity:

```
@Override
public void onPause() {
 if (mAdView != null) {
 mAdView.pause();
```

```
 }
 super.onPause();
}

@Override
public void onResume() {
 super.onResume();
 if (mAdView != null) {
 mAdView.resume();
 }
}

@Override
public void onDestroy() {
 if (mAdView != null) {
 mAdView.destroy();
 }
 super.onDestroy();
}
```

```
 }
 super.onPause();
}

@Override
public void onResume() {
 super.onResume();
 if (mAdView != null) {
 mAdView.resume();
 }
}

@Override
public void onDestroy() {
 if (mAdView != null) {
 mAdView.destroy();
 }
 super.onDestroy();
}
```

# 第243章：Google Play商店

## 第243.1节：打开应用的Google Play商店页面

以下代码片段展示了如何以安全的方式打开应用的Google Play商店页面。通常在请求用户为应用留下评价时使用。

```
private void openPlayStore() {
 String packageName = getPackageName();
 Intent playStoreIntent = new Intent(Intent.ACTION_VIEW,
 Uri.parse("market://details?id=" + packageName));
 setFlags(playStoreIntent);
 try {
 startActivity(playStoreIntent);
 } catch (Exception e) {
 Intent webIntent = new Intent(Intent.ACTION_VIEW,
 Uri.parse("https://play.google.com/store/apps/details?id=" + packageName));
 setFlags(webIntent);
 startActivity(webIntent);
 }
}

@SuppressWarnings("deprecation")
private void setFlags(Intent intent) {
 intent.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
 intent.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
 else
 intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
```

注意：如果应用已安装，代码将打开Google Play商店。否则，它将仅打开网页浏览器。

## 第243.2节：打开Google Play商店，显示您发布者账户中的所有应用列表

您可以在您的应用中添加一个“浏览我们的其他应用”按钮，列出您在Google Play商店应用中的所有（发布者）应用。

```
String urlApp = "market://search?q=pub:Google+Inc.";
String urlWeb = "http://play.google.com/store/search?q=pub:Google+Inc.";
try {
 Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlApp));
 setFlags(i);
 startActivity(i);
} catch (android.content.ActivityNotFoundException anfe) {
 Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlWeb));
 setFlags(i);
 startActivity(i);
}

@SuppressWarnings("deprecation")
public void setFlags(Intent i) {
 i.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
 i.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
 }
}
```

# Chapter 243: Google Play Store

## Section 243.1: Open Google Play Store Listing for your app

The following code snippet shows how to open the Google Play Store Listing of your app in a safe way. Usually you want to use it when asking the user to leave a review for your app.

```
private void openPlayStore() {
 String packageName = getPackageName();
 Intent playStoreIntent = new Intent(Intent.ACTION_VIEW,
 Uri.parse("market://details?id=" + packageName));
 setFlags(playStoreIntent);
 try {
 startActivity(playStoreIntent);
 } catch (Exception e) {
 Intent webIntent = new Intent(Intent.ACTION_VIEW,
 Uri.parse("https://play.google.com/store/apps/details?id=" + packageName));
 setFlags(webIntent);
 startActivity(webIntent);
 }
}

@SuppressWarnings("deprecation")
private void setFlags(Intent intent) {
 intent.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
 intent.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
 else
 intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
```

**Note:** The code opens the Google Play Store if the app is installed. Otherwise it will just open the web browser.

## Section 243.2: Open Google Play Store with the list of all applications from your publisher account

You can add a "Browse Our Other Apps" button in your app, listing all your(publisher) applications in the Google Play Store app.

```
String urlApp = "market://search?q=pub:Google+Inc.";
String urlWeb = "http://play.google.com/store/search?q=pub:Google+Inc.";
try {
 Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlApp));
 setFlags(i);
 startActivity(i);
} catch (android.content.ActivityNotFoundException anfe) {
 Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlWeb));
 setFlags(i);
 startActivity(i);
}

@SuppressWarnings("deprecation")
public void setFlags(Intent i) {
 i.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
 i.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
 }
}
```

```
else {
 i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
}
```

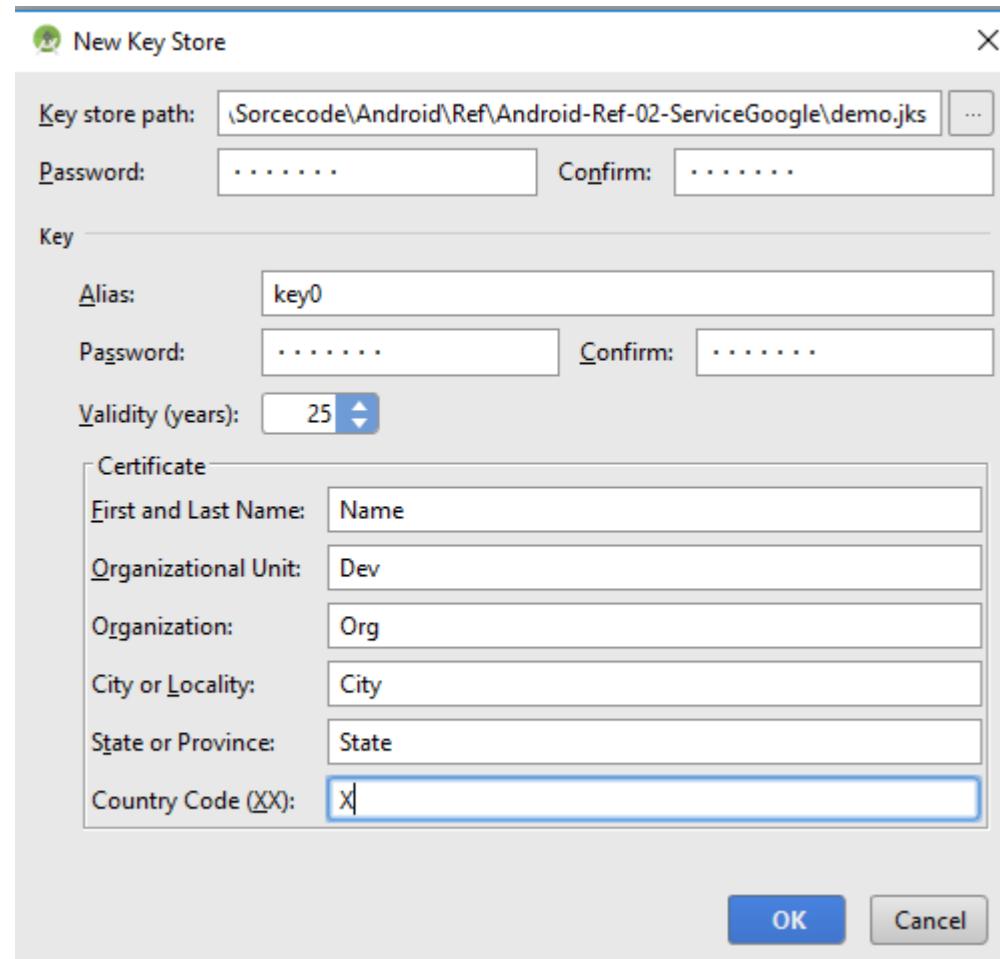
```
else {
 i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
}
```

# 第244章：为发布签署您的Android应用

Android要求所有APK在发布前必须签名。

## 第244.1节：为您的应用签名

1. 在菜单栏中，点击 构建 > 生成签名的APK。
2. 从下拉菜单中选择您想要发布的模块，然后点击下一步。
3. 要创建一个新的密钥库，点击“创建新建”。现在填写所需信息，然后在“新建密钥库”中按确定。

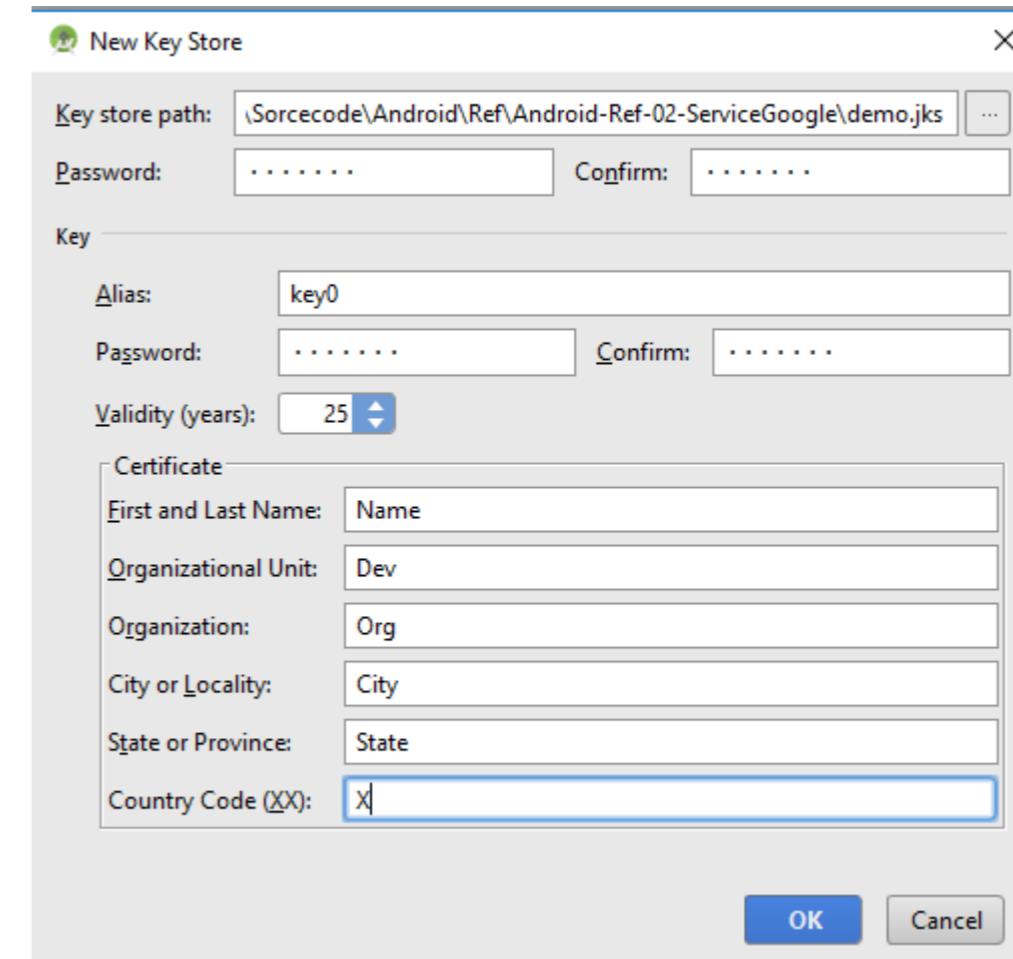


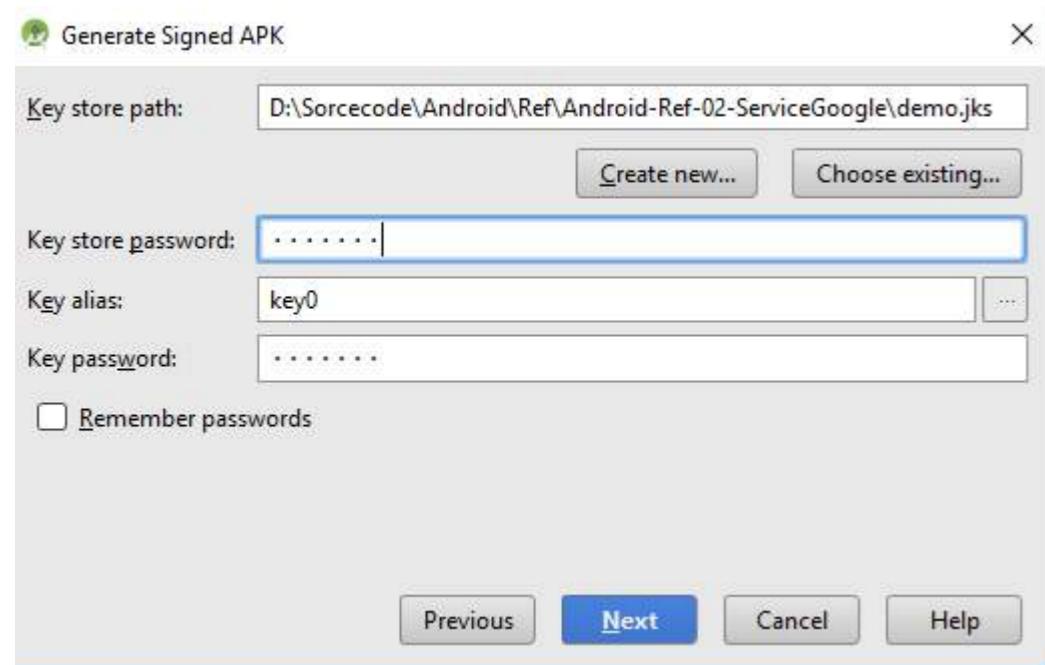
# Chapter 244: Sign your Android App for Release

Android requires that all APKs be signed for release.

## Section 244.1: Sign your App

1. In the menu bar, click Build > Generate Signed APK.
2. Select the module you would like to release from the drop down and click Next.
3. To Create a new keystore, click Create new. Now fill the required information and press ok in New Key Store.





4. 如果您刚创建了新的密钥库，生成签名APK向导中的字段会自动填充，否则请填写并点击下一步。

5. 在下一个窗口中，选择签名APK的保存位置，选择构建类型，然后点击完成。

## 第244.2节：配置build.gradle中的签名配置

您可以在build.gradle文件中定义签名配置来签署apk。

您可以定义：

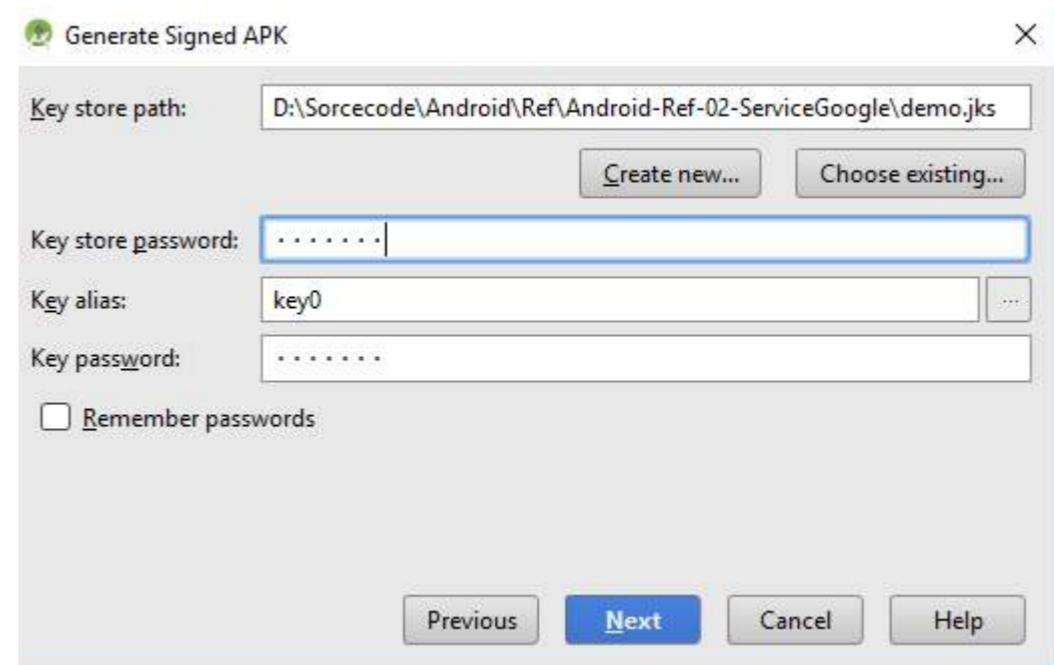
- storeFile : 密钥库文件
- storePassword : 密钥库密码
- keyAlias : 密钥别名
- keyPassword : 密钥别名密码

您必须定义signingConfigs块来创建签名配置：

```
android {
 signingConfigs {
 myConfig {
 storeFile file("myFile.keystore")
 storePassword "xxxx"
 keyAlias "xxxx"
 keyPassword "xxxx"
 }
 //...
}
```

然后你可以将其分配给一个或多个构建类型。

```
android {
 buildTypes {
```



4. On the Generate Signed APK Wizard fields are already populated for you if you just created new key store otherwise fill it and click next.

5. On the next window, select a destination for the signed APK, select the build type and click finish.

## Section 244.2: Configure the build.gradle with signing configuration

You can define the signing configuration to sign the apk in the build.gradle file.

You can define:

- storeFile : the keystore file
- storePassword: the keystore password
- keyAlias: a key alias name
- keyPassword: A key alias password

You have to **define** the signingConfigs block to create a signing configuration:

```
android {
 signingConfigs {
 myConfig {
 storeFile file("myFile.keystore")
 storePassword "xxxx"
 keyAlias "xxxx"
 keyPassword "xxxx"
 }
 //...
}
```

Then you can **assign** it to one or more build types.

```
android {
 buildTypes {
```

```
release {
 signingConfig signingConfigs.myConfig
}
}
```

```
release {
 signingConfig signingConfigs.myConfig
}
}
```

# 第245章：TensorFlow

TensorFlow 是专为移动和嵌入式平台设计的。我们提供了示例代码和构建支持，您现在可以在这些平台上试用：

Android iOS 树莓派

## 第245.1节：使用方法

从 [here](#) 安装 Bazel。Bazel 是 TensorFlow 的主要构建系统。现在，编辑 WORKSPACE，我们可以在之前克隆的 TensorFlow 根目录中找到 WORKSPACE 文件。

```
取消注释并更新这些条目中的路径以构建 Android 演示。
#android_sdk_repository(
name = "androidsdk",
api_level = 23,
build_tools_version = "25.0.1",
替换为系统中 Android SDK 的路径
path = "<PATH_TO_SDK>",
#)

#android_ndk_repository(
name="androidndk",
path="<PATH_TO_NDK>",
api_level=14)
```

如下所示，带有我们的 SDK 和 NDK 路径：

```
android_sdk_repository(
 name = "androidsdk",
 api_level = 23,
 build_tools_version = "25.0.1",
 # 请替换为您系统中 Android SDK 的路径
 path = "/Users/amitshekhar/Library/Android/sdk/",
)
android_ndk_repository(
 name="androidndk",
 path="/Users/amitshekhar/Downloads/android-ndk-r13/",
 api_level=14)
```

# Chapter 245: TensorFlow

TensorFlow was designed with mobile and embedded platforms in mind. We have sample code and build support you can try now for these platforms:

Android iOS Raspberry Pi

## Section 245.1: How to use

Install Bazel from [here](#). Bazel is the primary build system for TensorFlow. Now, edit the WORKSPACE, we can find the WORKSPACE file in the root directory of the TensorFlow that we have cloned earlier.

```
Uncomment and update the paths in these entries to build the Android demo.
#android_sdk_repository(
name = "androidsdk",
api_level = 23,
build_tools_version = "25.0.1",
Replace with path to Android SDK on your system
path = "<PATH_TO_SDK>",
#)

#android_ndk_repository(
name="androidndk",
path="<PATH_TO_NDK>",
api_level=14)
```

Like below with our sdk and ndk path:

```
android_sdk_repository(
 name = "androidsdk",
 api_level = 23,
 build_tools_version = "25.0.1",
 # Replace with path to Android SDK on your system
 path = "/Users/amitshekhar/Library/Android/sdk/",
)
android_ndk_repository(
 name="androidndk",
 path="/Users/amitshekhar/Downloads/android-ndk-r13/",
 api_level=14)
```

# 第246章：Android Vk SDK

## 第246.1节：初始化和登录

- 在此创建新应用程序：创建应用程序
- 选择独立应用程序并通过短信确认应用创建。
- 填写Android的包名为您当前的包名。您可以在Android的清单文件开头处获取您的包名。
- 通过在shell/cmd中执行以下命令获取您的证书指纹：

```
keytool -exportcert -alias androiddebugkey -keystore 路径-to-debug-or-production-keystore -list -v
```

您也可以通过SDK本身获取此指纹：

```
String[] fingerprints = VKUtil.getCertificateFingerprint(this, this.getPackageName());
Log.d("MainActivity", fingerprints[0]);
```

- 将收到的指纹添加到Vk应用设置中的Android签名证书指纹：字段  
(在您输入包名的地方)
- 然后将此添加到您的 gradle 文件中：

```
compile 'com.vk:androidsdk:1.6.5'
```

- 使用以下方法在启动时初始化 SDK。最好的方式是在 Application 的 onCreate 方法中调用它。

```
private static final int VK_ID = your_vk_id;
public static final String VK_API_VERSION = "5.52"; //当前版本
@Override
 public void onCreate() {
 super.onCreate();
 VKSdk.customInitialize(this, VK_ID, VK_API_VERSION);
 }
```

这是初始化 VKSdk 的最佳方式。不要使用将 VK\_ID 放在 strings.xml 中的方法，因为这样 API 将无法正常工作。

- 最后一步是使用 vksdk 登录。

```
public static final String[] VK_SCOPES = new String[]{
 VKScope.FRIENDS,
 VKScope.MESSAGES,
 VKScope.通知,
 VKScope.离线,
 VKScope.状态,
 VKScope.统计,
 VKScope.照片
};

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 someButtonForLogin.设置点击监听器(new View.OnClickListener() {
 @Override
 public void 点击事件(View v) {
 // 处理点击逻辑
 }
 });
}
```

# Chapter 246: Android Vk Sdk

## Section 246.1: Initialization and login

- Create a new application here: [create application](#)
- Choose standalone applicaton and confirm app creation via SMS.
- Fill **Package name for Android** as your current package name. You can get your package name inside android manifest file, at the very beginning.
- Get your **Certificate fingerprint** by executing this command in your shell/cmd:

```
keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -list -v
```

You can also get this fingerprint by SDK itself:

```
String[] fingerprints = VKUtil.getCertificateFingerprint(this, this.getPackageName());
Log.d("MainActivity", fingerprints[0]);
```

- Add received fingerprint into your **Signing certificate fingerprint for Android**: field in Vk app settings (where you entered your package name)
- Then add this to your gradle file:

```
compile 'com.vk:androidsdk:1.6.5'
```

- Initialize the SDK on startup using the following method. The best way is to call it in the Applications onCreate method.

```
private static final int VK_ID = your_vk_id;
public static final String VK_API_VERSION = "5.52"; //current version
@Override
 public void onCreate() {
 super.onCreate();
 VKSdk.customInitialize(this, VK_ID, VK_API_VERSION);
 }
```

This is the best way to initizlize VKSdk. Don't use the methid where VK\_ID should be placed inside strings.xml because api will not work correctly after it.

- Final step is to login using vksdk.

```
public static final String[] VK_SCOPES = new String[]{
 VKScope.FRIENDS,
 VKScope.MESSAGES,
 VKScope.NOTIFICATIONS,
 VKScope.OFFLINE,
 VKScope.STATUS,
 VKScope.STATS,
 VKScope.PHOTOS
};

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 someButtonForLogin.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
 // 处理点击逻辑
 }
 });
}
```

```
VKSdk.登录(this, VK_SCOPES);
 }
});

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
VKSdk.处理活动结果(requestCode, resultCode, data, new VKCallback<VKAccessToken>() {
 @Override
 public void 结果处理(VKAccessToken res) {
 res.accessToken; //在这里获取我们的令牌。
 }

 @Override
 public void 错误处理(VKError error) {
Toast.显示提示(SocialNetworkChooseActivity.this,
 "用户未通过授权", Toast.短时显示).show();
 }
});
}
```

```
VKSdk.login(this, VK_SCOPES);
 }
});

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
VKSdk.onActivityResult(requestCode, resultCode, data, new VKCallback<VKAccessToken>() {
 @Override
 public void onResult(VKAccessToken res) {
 res.accessToken; //getting our token here.
 }

 @Override
 public void onError(VKError error) {
 Toast.makeText(SocialNetworkChooseActivity.this,
 "User didn't pass Authorization", Toast.LENGTH_SHORT).show();
 }
});
```

# 第247章：项目SDK版本

参数	详细信息
SDK版本 每个字段的SDK版本是Android发布的SDK API级别整数。例如，Froyo (Android 2.2) 对应API级别8。这些整数也在Build.VERSION_CODES中定义。	

一个Android应用需要在各种设备上运行。每个设备上运行的Android版本可能不同。

现在，每个Android版本可能不支持您的应用所需的所有功能，因此在构建应用时，您需要考虑最低和最高的Android版本。

## 第247.1节：定义项目SDK版本

在主模块（app）的build.gradle文件中，定义您的最低和目标版本号。

```
android {
 //用于编译项目的SDK源版本
 compileSdkVersion 23

 defaultConfig {
 //设备运行您的应用所需的最低SDK版本
 minSdkVersion 19
 //通常你不需要设置最大SDK限制，这样你的应用可以支持未来版本的
 //安卓系统而无需更新应用
 //maxSdkVersion 23
 //
 //你针对（构建和测试）应用的安卓最新SDK版本，应该与compileSdkVersion相同
 targetSdkVersion 23
 }
}
```

# Chapter 247: Project SDK versions

Parameter	Details
SDK Version	The SDK version for each field is the Android release's SDK API level integer. For example, Froyo (Android 2.2) corresponds to API level 8. These integers are also defined in <a href="#">Build.VERSION_CODES</a> .

An Android application needs to run on all kinds of devices. Each device may have a different version of Android running on it.

Now, each Android version might not support all the features that your app requires, and so while building an app, you need to keep the minimum and maximum Android version in mind.

## Section 247.1: Defining project SDK versions

In your build.gradle file of main module(**app**), define your minimum and target version number.

```
android {
 //the version of sdk source used to compile your project
 compileSdkVersion 23

 defaultConfig {
 //the minimum sdk version required by device to run your app
 minSdkVersion 19
 //you normally don't need to set max sdk limit so that your app can support future versions
 //of android without updating app
 //maxSdkVersion 23
 //
 //the latest sdk version of android on which you are targeting(building and testing) your
 //app, it should be same as compileSdkVersion
 targetSdkVersion 23
 }
}
```

# 第248章：Android的Facebook SDK

参数	详细信息
标签	用于日志记录的字符串
FacebookSignInHelper	一个对Facebook助手的静态引用
CallbackManager	Facebook操作的回调
活动	上下文
PERMISSION_LOGIN	包含所有 Facebook 登录所需权限的数组。
loginCallback	Facebook 登录的回调函数

## 第 248.1 节：如何在 Android 中添加 Facebook 登录

在你的build.gradle中添加以下依赖项

```
// Facebook 登录
compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

将以下辅助类添加到您的工具包中：

```
/**
 * 由安迪创建
 * 一个用于Facebook的工具
 */
public class FacebookSignInHelper {
 private static final String TAG = FacebookSignInHelper.class.getSimpleName();
 private static FacebookSignInHelper facebookSignInHelper = null;
 private CallbackManager callbackManager;
 private Activity mActivity;
 private static final Collection<String> PERMISSION_LOGIN = (Collection<String>)
 Arrays.asList("public_profile", "user_friends", "email");
 private FacebookCallback<LoginResult> loginCallback;

 public static FacebookSignInHelper newInstance(Activity context) {
 if (facebookSignInHelper == null)
 facebookSignInHelper = new FacebookSignInHelper(context);
 return facebookSignInHelper;
 }

 public FacebookSignInHelper(Activity mActivity) {
 try {
 this.mActivity = mActivity;
 // 在执行任何其他操作之前初始化SDK,
 // 尤其是当你使用Facebook UI元素时。
 FacebookSdk.sdkInitialize(this.mActivity);
 callbackManager = CallbackManager.Factory.create();
 loginCallback = new FacebookCallback<LoginResult>() {
 @Override
 public void onSuccess(LoginResult loginResult) {
 // 你已登录Facebook
 }

 @Override
 public void onCancel() {
 Log.d(TAG, "Facebook: 用户取消");
 }
 };
 } catch (Exception e) {
 Log.e(TAG, "Facebook: 登录失败");
 }
 }

 public void logIn() {
 LoginManager.getInstance().logInWithReadPermissions(mActivity, PERMISSION_LOGIN);
 LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
 }

 public void logOut() {
 LoginManager.getInstance().logOut();
 }
}
```

# Chapter 248: Facebook SDK for Android

Parameter	Details
TAG	A String used while logging
FacebookSignInHelper	A static reference to facebook helper
CallbackManager	A callback for facebook operations
Activity	A context
PERMISSION_LOGIN	An array that contains all permission required from facebook to login.
loginCallback	A callback for facebook login

## Section 248.1: How to add Facebook Login in Android

Add below dependencies to your build.gradle

```
// Facebook login
compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

Add below helper class to your utility package:

```
/**
 * Created by Andy
 * An utility for Facebook
 */
public class FacebookSignInHelper {
 private static final String TAG = FacebookSignInHelper.class.getSimpleName();
 private static FacebookSignInHelper facebookSignInHelper = null;
 private CallbackManager callbackManager;
 private Activity mActivity;
 private static final Collection<String> PERMISSION_LOGIN = (Collection<String>)
 Arrays.asList("public_profile", "user_friends", "email");
 private FacebookCallback<LoginResult> loginCallback;

 public static FacebookSignInHelper newInstance(Activity context) {
 if (facebookSignInHelper == null)
 facebookSignInHelper = new FacebookSignInHelper(context);
 return facebookSignInHelper;
 }

 public FacebookSignInHelper(Activity mActivity) {
 try {
 this.mActivity = mActivity;
 // Initialize the SDK before executing any other operations,
 // especially, if you're using Facebook UI elements.
 FacebookSdk.sdkInitialize(this.mActivity);
 callbackManager = CallbackManager.Factory.create();
 loginCallback = new FacebookCallback<LoginResult>() {
 @Override
 public void onSuccess(LoginResult loginResult) {
 // You are logged into Facebook
 }

 @Override
 public void onCancel() {
 Log.d(TAG, "Facebook: Cancelled by user");
 }
 };
 } catch (Exception e) {
 Log.e(TAG, "Facebook: Login failed");
 }
 }

 public void logIn() {
 LoginManager.getInstance().logInWithReadPermissions(mActivity, PERMISSION_LOGIN);
 LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
 }

 public void logOut() {
 LoginManager.getInstance().logOut();
 }
}
```

```

 }

 @Override
 public void onError(FacebookException error) {
 Log.d(TAG, "Facebook异常: " + error.getMessage());
 }
}

} 捕获 (异常 e) {
e.printStackTrace();
}

/***
* 在没有默认Facebook按钮的情况下登录用户
*/
public void loginUser() {
 try {
LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
 LoginManager.getInstance().logInWithReadPermissions(this.mActivity, PERMISSION_LOGIN);
 } catch (Exception e) {
e.printStackTrace();
 }
}

/***
* 使用用户从 Facebook 登出
*/
public void signOut() {
 // Facebook 登出
LoginManager.getInstance().logOut();
}

public CallbackManager getCallbackManager() {
 return callbackManager;
}

public FacebookCallback<LoginResult> getLoginCallback() {
 return loginCallback;
}

/***
* 尝试记录 Facebook 的调试密钥哈希
*/
* @param context : 上下文引用
* @return : Facebook 调试密钥哈希
*/
public static String getKeyHash(Context context) {
 String keyHash = null;
 try {
PackageManager info = context.getPackageManager().getPackageInfo(
 context.getPackageName(),
PackageManager.GET_SIGNATURES);
 for (Signature signature : info.signatures) {
 MessageDigest md = MessageDigest.getInstance("SHA");
 md.update(signature.toByteArray());
 keyHash = Base64.encodeToString(md.digest(), Base64.DEFAULT);
 Log.d(TAG, "KeyHash:" + keyHash);
 }
 } catch (PackageManager.NameNotFoundException e) {
e.printStackTrace();
 } catch (NoSuchAlgorithmException e) {
}
}


```

```

 }

 @Override
 public void onError(FacebookException error) {
 Log.d(TAG, "FacebookException: " + error.getMessage());
 }
}

} catch (Exception e) {
e.printStackTrace();
}

/***
* To login user on facebook without default Facebook button
*/
public void loginUser() {
 try {
 LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
 LoginManager.getInstance().logInWithReadPermissions(this.mActivity, PERMISSION_LOGIN);
 } catch (Exception e) {
e.printStackTrace();
 }
}

/***
* To log out user from facebook
*/
public void signOut() {
 // Facebook sign out
 LoginManager.getInstance().logOut();
}

public CallbackManager getCallbackManager() {
 return callbackManager;
}

public FacebookCallback<LoginResult> getLoginCallback() {
 return loginCallback;
}

/***
* Attempts to log debug key hash for facebook
*/
* @param context : A reference to context
* @return : A facebook debug key hash
*/
public static String getKeyHash(Context context) {
 String keyHash = null;
 try {
 PackageManager info = context.getPackageManager().getPackageInfo(
 context.getPackageName(),
PackageManager.GET_SIGNATURES);
 for (Signature signature : info.signatures) {
 MessageDigest md = MessageDigest.getInstance("SHA");
 md.update(signature.toByteArray());
 keyHash = Base64.encodeToString(md.digest(), Base64.DEFAULT);
 Log.d(TAG, "KeyHash:" + keyHash);
 }
 } catch (PackageManager.NameNotFoundException e) {
e.printStackTrace();
 } catch (NoSuchAlgorithmException e) {
}
}


```

```

e.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
}
return keyHash;
}
}

```

在您的活动中添加以下代码：

```

FacebookSignInHelper facebookSignInHelper = FacebookSignInHelper.newInstance(LoginActivity.this,
fireBaseAuthHelper);
facebookSignInHelper.loginUser();

```

在您的OnActivityResult中添加以下代码：

```

facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);

```

## 第248.2节：为Facebook登录创建您自己的自定义按钮

一旦您首次添加了Facebook登录/注册，按钮看起来大致如下：



大多数情况下，它与您的应用设计规范不匹配。以下是您可以自定义的方法：

```

<FrameLayout
 android:layout_below="@+id/no_network_bar"
 android:id="@+id/Layout1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <com.facebook.login.widget.LoginButton
 android:id="@+id/login_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:visibility="gone" />

 <Button
 android:background="#3B5998"
 android:layout_width="match_parent"
 android:layout_height="60dp"
 android:id="@+id/fb"
 android:onClick="onClickFacebookButton"
 android:textAllCaps="false"
 android:text="使用 Facebook 注册"
 android:textSize="22sp"
 android:textColor="#ffffff" />
</FrameLayout>

```

只需将原始的com.facebook.login.widget.LoginButton包裹在一个FrameLayout中，并将其可见性设置为gone。

接下来，在同一个FrameLayout中添加你的自定义按钮。我添加了一些示例规格。你也可以自己制作facebook按钮的drawable背景，并将其设置为按钮的背景。

```

e.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
}
return keyHash;
}
}

```

Add below code in Your Activity:

```

FacebookSignInHelper facebookSignInHelper = FacebookSignInHelper.newInstance(LoginActivity.this,
fireBaseAuthHelper);
facebookSignInHelper.loginUser();

```

Add below code to your OnActivityResult:

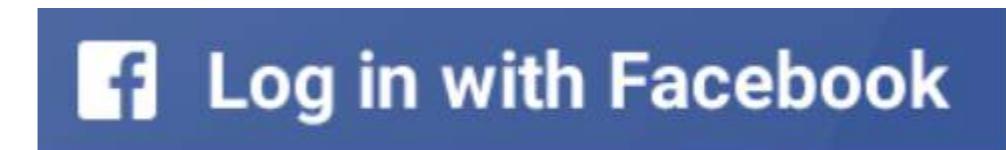
```

facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);

```

## Section 248.2: Create your own custom button for Facebook login

Once you first add the Facebook login/signup, the button looks something like:



Most of the times, it doesn't match with the design-specs of your app. And here's how you can customize it:

```

<FrameLayout
 android:layout_below="@+id/no_network_bar"
 android:id="@+id/Layout1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <com.facebook.login.widget.LoginButton
 android:id="@+id/login_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:visibility="gone" />

 <Button
 android:background="#3B5998"
 android:layout_width="match_parent"
 android:layout_height="60dp"
 android:id="@+id/fb"
 android:onClick="onClickFacebookButton"
 android:textAllCaps="false"
 android:text="Sign up with Facebook"
 android:textSize="22sp"
 android:textColor="#ffffff" />
</FrameLayout>

```

Just wrap the original com.facebook.login.widget.LoginButton into a FrameLayout and make its visibility gone.

Next, add your custom button in the same FrameLayout. I've added some sample specs. You can always make your own drawable background for the facebook button and set it as the background of the button.

最后，我们所做的就是将我自定义按钮的点击事件转换为facebook按钮的点击事件：

```
//原始的Facebook按钮
LoginButton loginButton = (LoginButton)findViewByI(R.id.login_button);

//我们的自定义Facebook按钮
fb = (Button) findViewByI(R.id.fb);

public void onClickFacebookButton(View view) {
 if (view == fb) {
 loginButton.performClick();
 }
}
```

太好了！现在按钮看起来像这样：



## 第248.3节：Facebook登录/注册的极简指南实现

1. 你必须设置先决条件。
2. 将Facebook活动添加到AndroidManifest.xml文件中：

```
<activity
 android:name="com.facebook.FacebookActivity"
 android:configChanges= "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
 android:theme="@android:style/Theme.Translucent.NoTitleBar"
 android:label="@string/app_name" />
```

3. 将登录按钮添加到你的布局XML文件中：

```
<com.facebook.login.widget.LoginButton
 android:id="@+id/login_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
```

4. 现在你有了Facebook按钮。如果用户点击它，Facebook登录对话框将会出现在应用程序屏幕的顶部。用户可以在这里填写他们的凭证并按下登录按钮。如果凭证正确，对话框将授予相应的权限，并向包含该按钮的原始活动发送回调。以下代码展示了如何接收该回调：

```
loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
 @Override
 public void onSuccess(LoginResult loginResult) {
 // 完成且无错误。你可能想在这里使用检索到的数据。
 }

 @Override
 public void onCancel() {
 // 用户要么取消了 Facebook 登录流程，要么未授权该应用。
 }

 @Override
```

The final thing we do is simply convert the click on my custom button to a click on the facebook button:

```
//The original Facebook button
LoginButton loginButton = (LoginButton)findViewByI(R.id.login_button);

//Our custom Facebook button
fb = (Button) findViewByI(R.id.fb);

public void onClickFacebookButton(View view) {
 if (view == fb) {
 loginButton.performClick();
 }
}
```

Great! Now the button looks something like this:



## Section 248.3: A minimalistic guide to Facebook login/signup implementation

1. You have to setup the [prerequisites](#).
2. Add the Facebook activity to the *AndroidManifest.xml* file:

```
<activity
 android:name="com.facebook.FacebookActivity"
 android:configChanges= "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
 android:theme="@android:style/Theme.Translucent.NoTitleBar"
 android:label="@string/app_name" />
```

3. Add the login button to your layout XML file:

```
<com.facebook.login.widget.LoginButton
 android:id="@+id/login_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
```

4. Now you have the Facebook button. If the user clicks on it, the Facebook login dialog will come up on top of the app's screen. Here the user can fill in their credentials and press the *Log In* button. If the credentials are correct, the dialog grants the corresponding permissions and a callback is sent to your original activity containing the button. The following code shows how you can receive that callback:

```
loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
 @Override
 public void onSuccess(LoginResult loginResult) {
 // Completed without error. You might want to use the retrieved data here.
 }

 @Override
 public void onCancel() {
 // The user either cancelled the Facebook login process or didn't authorize the app.
 }

 @Override
```

```
public void onError(FacebookException exception) {
 // 对话框因错误关闭。异常信息将帮助你识别具体出了什么问题。
}
});
```

```
public void onError(FacebookException exception) {
 // The dialog was closed with an error. The exception will help you recognize what
 // exactly went wrong.
}
});
```

## 第248.4节：设置权限以访问Facebook资料中的数据

如果你想获取用户的Facebook资料详情，需要为此设置权限：

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

你可以继续添加更多权限，如好友列表、帖子、照片等。只需选择合适的权限并添加到上述列表中即可。

注意：访问公开资料（名字、姓氏、ID、性别等）无需设置任何显式权限。

## Section 248.4: Setting permissions to access data from the Facebook profile

If you want to retrieve the details of a user's Facebook profile, you need to set permissions for the same:

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

You can keep adding more permissions like friends-list, posts, photos etc. Just pick [the right permission](#) and add it the above list.

Note: You don't need to set any explicit permissions for accessing the public profile (first name, last name, id, gender etc).

## 第248.5节：退出Facebook

从 Facebook SDK 4.0 开始，登出的方法如下：

```
com.facebook.login.LoginManager.getInstance().logOut();
```

对于 4.0 之前的版本，登出是通过显式清除访问令牌来实现的：

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

## Section 248.5: Logging out of Facebook

Facebook SDK 4.0 onwards, this is how we logout:

```
com.facebook.login.LoginManager.getInstance().logOut();
```

For versions before 4.0, the logging out is done by explicitly clearing the access token:

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

# 第 249 章：线程

## 第 249.1 节：线程示例及其说明

启动应用程序时，首先执行主线程。这个主线程处理应用程序的所有用户界面（UI）相关内容。如果我们想运行一个不需要 UI 的长时间任务，则使用线程在后台运行该任务。

下面是一个描述如下的线程示例：

```
new Thread(new Runnable() {
 public void run() {
 for(int i = 1; i < 5;i++) {
 System.out.println(i);
 }
 }
}).start();
```

我们可以通过创建 Thread 对象来创建线程，该对象具有用于运行线程的 Thread.run() 方法。这里，run() 方法由 start() 方法调用。

我们也可以独立运行多个线程，这称为多线程（MultiThreading）。该线程还具有 sleep 功能，可以使当前执行的线程休眠（暂时停止执行）指定的时间。但 sleep 会抛出 InterruptedException，因此我们必须使用 try/catch 来处理，如下所示。

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

## 第249.2节：从后台线程更新UI

通常会使用后台线程执行网络操作或长时间运行的任务，然后在需要时用结果更新UI。

这会带来一个问题，因为只有主线程可以更新UI。

解决方案是使用 runOnUiThread() 方法，因为它允许你从后台线程启动代码在UI线程上执行。

在这个简单的例子中，当 Activity 创建时启动一个线程，该线程运行直到随机生成魔术数字 42，然后使用 runOnUiThread() 方法在满足条件时更新UI。

```
public class MainActivity extends AppCompatActivity {

 TextView mTextView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 mTextView = (TextView) findViewById(R.id.my_text_view);

 new Thread(new Runnable() {
 @Override
 public void run() {
 while (true) {
```

# Chapter 249: Thread

## Section 249.1: Thread Example with its description

While launching an application firstly main thread is executed. This Main thread handles all the UI concept of application. If we want to run long the task in which we don't need the UI then we use thread for running that task in background.

**Here is the example of Thread which describes blow:**

```
new Thread(new Runnable() {
 public void run() {
 for(int i = 1; i < 5;i++) {
 System.out.println(i);
 }
 }
}).start();
```

We can create thread by creating the object of Thread which have [Thread.run\(\)](#) method for running the thread. Here, run() method is called by the start() method.

We can also run the multiple threads independently, which is known as MultiThreading. This thread also have the functionality of sleep by which the currently executing thread to sleep (temporarily cease execution) for the specified number of time. But sleep throws the InterruptedException So, we have to handle it by using try/catch like this.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

## Section 249.2: Updating the UI from a Background Thread

It is common to use a background Thread for doing network operations or long running tasks, and then update the UI with the results when needed.

This poses a problem, as only the main thread can update the UI.

The solution is to use the [runOnUiThread\(\)](#) method, as it allows you to initiate code execution on the UI thread from a background Thread.

In this simple example, a Thread is started when the Activity is created, runs until the magic number of 42 is randomly generated, and then uses the runOnUiThread() method to update the UI once this condition is met.

```
public class MainActivity extends AppCompatActivity {

 TextView mTextView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 mTextView = (TextView) findViewById(R.id.my_text_view);

 new Thread(new Runnable() {
 @Override
 public void run() {
```

```
//执行操作....
Random r = new Random();
if (r.nextInt(100) == 42) {
 break;
}

runOnUiThread(new Runnable() {
 @Override
 public void run() {
mTextView.setText("Ready Player One");
 }
}).start();
}
```

```
//do stuff....
Random r = new Random();
if (r.nextInt(100) == 42) {
 break;
}

runOnUiThread(new Runnable() {
 @Override
 public void run() {
mTextView.setText("Ready Player One");
 }
}).start();
}
```

# 第250章：AsyncTask

参数	详细信息
参数	执行任务时传入的参数类型。
进度	后台计算期间发布的进度单元的类型
结果	后台计算结果的类型。

## 第250.1节：基本用法

在Android的[活动 \(Activities\)](#) 和[服务 \(Services\)](#) 中，大多数回调都在[主线程 \(main thread\)](#) 上运行。这使得更新UI变得简单，但在主线程上运行处理器或I/O密集型任务可能会导致UI暂停并变得无响应（[官方文档](#)中说明了随后会发生的情况）。

你可以通过将这些较重的任务放到后台线程来解决这个问题。

一种方法是使用[AsyncTask](#)，它提供了一个框架，方便地使用后台线程，并且可以在后台线程完成工作之前、期间和之后执行UI线程任务。

**扩展AsyncTask时可以重写的方法：**

- `onPreExecute()`：在任务执行前在UI线程上调用
- `doInBackground()`：在后台线程上调用，紧接着`onPreExecute()`执行完毕后。
- `onProgressUpdate()`：在调用`publishProgress(Progress...)`后，在UI线程上被调用。
- `onPostExecute()`：在后台计算完成后，在**UI线程**上调用

**示例**

```
public class MyCustomAsyncTask extends AsyncTask<File, Void, String> {

 @Override
 protected void onPreExecute(){
 // 这将在后台线程执行之前在UI线程上运行。
 super.onPreExecute();
 // 执行线程前的任务，如初始化变量。
 Log.v("myBackgroundTask", "Starting Background Task");
 }

 @Override
 protected String doInBackground(File... params) {
 // 磁盘密集型工作。这将在后台线程上运行。
 // 在文件中搜索包含"Hello"的第一行，并返回
 // 该行。
 try (Scanner scanner = new Scanner(params[0])) {
 while (scanner.hasNextLine()) {
 final String line = scanner.nextLine();
 publishProgress(); // 告诉UI线程我们取得了进展

 if (line.contains("Hello")) {
 return line;
 }
 }
 return null;
 }
 }
}
```

# Chapter 250: AsyncTask

Parameter	Details
Params	the type of the parameters sent to the task upon execution.
Progress	the type of the progress units published during the background computation
Result	the type of the result of the background computation.

## Section 250.1: Basic Usage

In Android [Activities](#) and [Services](#), most callbacks are run on the [main thread](#). This makes it simple to update the UI, but running processor- or I/O-heavy tasks on the main thread can cause your UI to pause and become unresponsive ([official documentation](#) on what then happens).

You can remedy this by putting these heavier tasks on a background thread.

One way to do this is using an [AsyncTask](#), which provides a framework to facilitate easy usage of a background Thread, and also perform UI Thread tasks before, during, and after the background Thread has completed its work.

**Methods that can be overridden when extending AsyncTask:**

- `onPreExecute()` : invoked on the **UI thread** before the task is executed
- `doInBackground()`: invoked on **the background thread** immediately after `onPreExecute()` finishes executing.
- `onProgressUpdate()`: invoked on the **UI thread** after a call to `publishProgress(Progress...)`.
- `onPostExecute()`: invoked on the **UI thread** after the background computation finishes

**Example**

```
public class MyCustomAsyncTask extends AsyncTask<File, Void, String> {

 @Override
 protected void onPreExecute(){
 // This runs on the UI thread before the background thread executes.
 super.onPreExecute();
 // Do pre-thread tasks such as initializing variables.
 Log.v("myBackgroundTask", "Starting Background Task");
 }

 @Override
 protected String doInBackground(File... params) {
 // Disk-intensive work. This runs on a background thread.
 // Search through a file for the first line that contains "Hello", and return
 // that line.
 try (Scanner scanner = new Scanner(params[0])) {
 while (scanner.hasNextLine()) {
 final String line = scanner.nextLine();
 publishProgress(); // tell the UI thread we made progress

 if (line.contains("Hello")) {
 return line;
 }
 }
 return null;
 }
 }
}
```

```

@Override
protected void onProgressUpdate(Void...p) {
 // 在调用publishProgress后在UI线程上运行
 Log.v("读取了另一行！");
}

@Override
protected void onPostExecute(String s) {
 // 该方法在doInBackground()方法执行完成后在UI线程上运行
 // 该函数接收doInBackground()方法返回的结果(String s)。
 // 使用找到的字符串更新UI。
TextView view = (TextView) findViewById(R.id.found_string);
 if (s != null) {
 view.setText(s);
 } else {
 view.setText("未找到匹配项。");
 }
}

```

### 用法：

```

MyCustomAsyncTask asyncTask = new MyCustomAsyncTask<File, Void, String>();
// 使用用户提供的文件名运行任务。
asyncTask.execute(userSuppliedFilename);

```

或者简单地：

```
new MyCustomAsyncTask().execute(userSuppliedFilename);
```

### 注意

定义AsyncTask时，我们可以在< >括号中传递三种类型。

定义为<Params, Progress, Result> (参见参数部分)

在前面的例子中，我们使用了类型<File, Void, String>：

```

AsyncTask<File, Void, String>
// Params的类型是File
// Progress未使用类型
// Result的类型是String

```

当你想标记某个类型未使用时，使用Void。

请注意，您不能将原始类型（即int、float及其他6种）作为参数传递。在这种情况下，您应该传递它们的包装类，例如使用Integer代替int，或使用Float代替float。

### AsyncTask与Activity生命周期

AsyncTask不遵循Activity实例的生命周期。如果您在Activity中启动AsyncTask并旋转设备，Activity将被销毁并创建一个新实例。但AsyncTask不会终止。它会继续运行直到完成。

### 解决方案：AsyncTaskLoader

Loaders的一个子类是AsyncTaskLoader。该类执行与AsyncTask相同的功能，但效果更好。它可以更轻松地处理Activity配置更改，并且其行为符合Fragment和Activity的生命周期。好处是AsyncTaskLoader可以在任何使用AsyncTask的场景中使用。每当需要将数据加载到内存供Activity/Fragment处理时，AsyncTaskLoader都能

```

@Override
protected void onProgressUpdate(Void...p) {
 // Runs on the UI thread after publishProgress is invoked
 Log.v("Read another line!");
}

@Override
protected void onPostExecute(String s) {
 // This runs on the UI thread after complete execution of the doInBackground() method
 // This function receives result(String s) returned from the doInBackground() method.
 // Update UI with the found string.
 TextView view = (TextView) findViewById(R.id.found_string);
 if (s != null) {
 view.setText(s);
 } else {
 view.setText("Match not found.");
 }
}

```

### Usage:

```

MyCustomAsyncTask asyncTask = new MyCustomAsyncTask<File, Void, String>();
// Run the task with a user supplied filename.
asyncTask.execute(userSuppliedFilename);

```

or simply:

```
new MyCustomAsyncTask().execute(userSuppliedFilename);
```

### Note

When defining an AsyncTask we can pass three types between < > brackets.

Defined as <Params, Progress, Result> (see **Parameters section**)

In the previous example we've used types <File, Void, String>:

```

AsyncTask<File, Void, String>
// Params has type File
// Progress has unused type
// Result has type String

```

**Void** is used when you want to mark a type as unused.

Note that you can't pass primitive types (i.e. **int**, **float** and 6 others) as parameters. In such cases, you should pass their wrapper classes, e.g. **Integer** instead of **int**, or **Float** instead of **float**.

### The AsyncTask and Activity life cycle

AsyncTasks don't follow Activity instances' life cycle. If you start an AsyncTask inside an Activity and you rotate the device, the Activity will be destroyed and a new instance will be created. But the AsyncTask will not die. It will go on living until it completes.

### Solution: AsyncTaskLoader

One subclass of Loaders is the **AsyncTaskLoader**. This class performs the same function as the **AsyncTask**, but much better. It can handle Activity configuration changes more easily, and it behaves within the life cycles of Fragments and Activities. The nice thing is that the **AsyncTaskLoader** can be used in any situation that the **AsyncTask** is being used. Anytime data needs to be loaded into memory for the Activity/Fragment to handle, The **AsyncTaskLoader** can

更好地完成任务。

## 第250.2节：将Activity作为WeakReference传递以避免内存泄漏

AsyncTask通常需要引用调用它的Activity。

如果AsyncTask是Activity的内部类，则可以直接引用Activity及其任何成员变量/方法。

但是，如果 AsyncTask 不是 Activity 的内部类，则需要将 Activity 的引用传递给 AsyncTask。当你这样做时，可能出现的一个问题是，AsyncTask 会一直持有 Activity 的引用，直到 AsyncTask 在其后台线程中完成工作。如果 Activity 在 AsyncTask 的后台线程工作完成之前被结束或销毁，AsyncTask 仍然会持有对该 Activity 的引用，因此该 Activity 无法被垃圾回收。

因此，这将导致内存泄漏。

为了防止这种情况发生，在 AsyncTask 中使用WeakReference，而不是直接引用 Activity。

下面是一个使用 WeakReference 的 AsyncTask 示例：

```
private class MyAsyncTask extends AsyncTask<String, Void, Void> {

 private WeakReference<Activity> mActivity;

 public MyAsyncTask(Activity activity) {
 mActivity = new WeakReference<Activity>(activity);
 }

 @Override
 protected void onPreExecute() {
 final Activity activity = mActivity.get();
 if (activity != null) {
 ...
 }
 }

 @Override
 protected Void doInBackground(String... params) {
 //执行某些操作
 String param1 = params[0];
 String param2 = params[1];
 return null;
 }

 @Override
 protected void onPostExecute(Void result) {
 final Activity activity = mActivity.get();
 if (activity != null) {
 activity.updateUI();
 }
 }
}
```

从 Activity 调用 AsyncTask :

do the job better.

## Section 250.2: Pass Activity as WeakReference to avoid memory leaks

It is common for an AsyncTask to require a reference to the Activity that called it.

If the AsyncTask is an inner class of the Activity, then you can reference it and any member variables/methods directly.

If, however, the AsyncTask is not an inner class of the Activity, you will need to pass an Activity reference to the AsyncTask. When you do this, one potential problem that may occur is that the AsyncTask will keep the reference of the Activity until the AsyncTask has completed its work in its background thread. If the Activity is finished or killed before the AsyncTask's background thread work is done, the AsyncTask will still have its reference to the Activity, and therefore it cannot be garbage collected.

As a result, this will cause a memory leak.

In order to prevent this from happening, make use of a [WeakReference](#) in the AsyncTask instead of having a direct reference to the Activity.

Here is an example AsyncTask that utilizes a WeakReference:

```
private class MyAsyncTask extends AsyncTask<String, Void, Void> {

 private WeakReference<Activity> mActivity;

 public MyAsyncTask(Activity activity) {
 mActivity = new WeakReference<Activity>(activity);
 }

 @Override
 protected void onPreExecute() {
 final Activity activity = mActivity.get();
 if (activity != null) {
 ...
 }
 }

 @Override
 protected Void doInBackground(String... params) {
 //Do something
 String param1 = params[0];
 String param2 = params[1];
 return null;
 }

 @Override
 protected void onPostExecute(Void result) {
 final Activity activity = mActivity.get();
 if (activity != null) {
 activity.updateUI();
 }
 }
}
```

Calling the AsyncTask from an Activity:

```
new MyAsyncTask(this).execute("param1", "param2");
```

从 Fragment 调用 AsyncTask :

```
new MyAsyncTask(getActivity()).execute("param1", "param2");
```

## 第 250.3 节：使用 AsyncTask 在 Android 中下载图片

本教程讲解如何使用 AsyncTask 在 Android 中下载图片。下面的示例在下载图片的同时显示进度条。理解 Android AsyncTask

AsyncTask 使您能够实现多线程编程，而无需直接操作线程。AsyncTask 使 UI 线程的使用变得正确且简单。它允许执行后台操作并将结果传递给 UI 线程。如果您正在做与 UI 相关的独立操作，例如下载数据以在列表中展示，可以使用 AsyncTask。

- AsyncTask 理想情况下应用于短时间操作（最多几秒）。异步任务由三个泛型类型定义，称为 Params、Progress 和 Result，以及四个步骤，称为 onPreExecute()、doInBackground()、onProgressUpdate() 和 onPostExecute()。
- 在onPreExecute()中，您可以定义需要在后台处理开始之前执行的代码。
- doInBackground 中有需要在后台执行的代码，这里在 doInBackground() 中，我们可以通过 publishProgress() 方法多次向事件线程发送结果，通知后台处理已完成，我们可以简单地返回结果。
- onProgressUpdate() 方法接收来自 doInBackground() 方法的进度更新，这些更新是通过 publishProgress() 方法发布的，该方法可以使用这些进度更新来更新事件线程。
- onPostExecute() 方法处理由 doInBackground() 方法返回的结果。
- 使用的泛型类型有
  - Params，执行任务时传入的参数类型
  - Progress，后台计算过程中发布的进度单位类型。
  - Result，后台计算结果的类型。
- 如果异步任务不使用任何类型，则可以标记为 Void 类型。
- 正在运行的异步任务可以通过调用 cancel(boolean) 方法取消。

使用 Android AsyncTask 下载图片

your .xml 布局

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical" >

 <Button
 android:id="@+id/downloadButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="点击此处下载" />

 <ImageView
 android:id="@+id/imageView"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:contentDescription="您的图片将显示在这里" />

```

```
new MyAsyncTask(this).execute("param1", "param2");
```

Calling the AsyncTask from a Fragment:

```
new MyAsyncTask(getActivity()).execute("param1", "param2");
```

## Section 250.3: Download Image using AsyncTask in Android

This tutorial explains how to download Image using AsyncTask in Android. The example below download image while showing progress bar while during download. **Understanding Android AsyncTask**  
Async task enables you to implement MultiThreading without get Hands dirty into threads. AsyncTask enables proper and easy use of the UI thread. It allows performing background operations and passing the results on the UI thread. If you are doing something isolated related to UI, for example downloading data to present in a list, go ahead and use AsyncTask.

- AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute(), doInBackground(), onProgressUpdate() and onPostExecute().
- In onPreExecute() you can define code, which need to be executed before background processing starts.
- doInBackground have code which needs to be executed in background, here in doInBackground() we can send results to multiple times to event thread by publishProgress() method, to notify background processing has been completed we can return results simply.
- onProgressUpdate() method receives progress updates from doInBackground() method, which is published via publishProgress() method, and this method can use this progress update to update event thread
- onPostExecute() method handles results returned by doInBackground() method.
- The generic types used are
  - Params, the type of the parameters sent to the task upon execution
  - Progress, the type of the progress units published during the background computation.
  - Result, the type of the result of the background computation.
- If an async task not using any types, then it can be marked as Void type.
- An running async task can be cancelled by calling cancel(boolean) method.

Downloading image using Android AsyncTask

your .xml layout

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical" >

 <Button
 android:id="@+id/downloadButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Click Here to Download" />

 <ImageView
 android:id="@+id/imageView"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:contentDescription="Your image will appear here" />

```

```
</LinearLayout>
```

### .java 类

```
package com.javatechig.droid;

import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class ImageDownladerActivity extends Activity {

 private ImageView downloadedImg;
 private ProgressDialog simpleWaitDialog;
 private String downloadUrl = "http://www.9ori.com/store/media/images/8ab579a656.jpg";

 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.async);
 Button imageDownloaderBtn = (Button) findViewById(R.id.downloadButton);

 downloadedImg = (ImageView) findViewById(R.id.imageView);
 imageDownloaderBtn.setOnClickListener(new OnClickListener() {

 @Override
 public void onClick(View v) {
 // TODO Auto-generated method stub
 new ImageDownloader().execute(downloadUrl);
 }
 });
 }

 private class ImageDownloader extends AsyncTask {

 @Override
 protected Bitmap doInBackground(String... param) {
 // TODO Auto-generated method stub
 return downloadBitmap(param[0]);
 }

 @Override
 protected void onPostExecute() {
 Log.i("Async-Example", "onPostExecute Called");
 simpleWaitDialog = ProgressDialog.show(ImageDownladerActivity.this,
```

```
</LinearLayout>
```

### .java class

```
package com.javatechig.droid;

import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class ImageDownladerActivity extends Activity {

 private ImageView downloadedImg;
 private ProgressDialog simpleWaitDialog;
 private String downloadUrl = "http://www.9ori.com/store/media/images/8ab579a656.jpg";

 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.async);
 Button imageDownloaderBtn = (Button) findViewById(R.id.downloadButton);

 downloadedImg = (ImageView) findViewById(R.id.imageView);
 imageDownloaderBtn.setOnClickListener(new OnClickListener() {

 @Override
 public void onClick(View v) {
 // TODO Auto-generated method stub
 new ImageDownloader().execute(downloadUrl);
 }
 });
 }

 private class ImageDownloader extends AsyncTask {

 @Override
 protected Bitmap doInBackground(String... param) {
 // TODO Auto-generated method stub
 return downloadBitmap(param[0]);
 }

 @Override
 protected void onPostExecute() {
 Log.i("Async-Example", "onPostExecute Called");
 simpleWaitDialog = ProgressDialog.show(ImageDownladerActivity.this,
```

```

 "等待", "正在下载图片");

}

@Override
protected void onPostExecute(Bitmap result) {
 Log.i("异步示例", "onPostExecute 被调用");
 downloadedImg.setImageBitmap(result);
simpleWaitDialog.dismiss();
}

private Bitmap downloadBitmap(String url) {
 // 初始化默认的 HTTP 客户端对象
 final DefaultHttpClient client = new DefaultHttpClient();

 // 形成一个 HttpGet 请求
 final HttpGet getRequest = new HttpGet(url);
 try {

HttpResponse response = client.execute(getRequest);

 // 检查 200 OK 是否成功
 final int statusCode = response.getStatusLine().getStatusCode();

 if (statusCode != HttpStatus.SC_OK) {
Log.w("ImageDownloader", "从 " + url + " 获取位图时出错, 状态码 " + statusCode)
 ;
 return null;
 }

 final HttpEntity entity = response.getEntity();
 if (entity != null) {
 InputStream inputStream = null;
 try {
 // 从流中获取内容
inputStream = entity.getContent();

 // 将流数据解码回安卓能识别的图像Bitmap
 final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

 return bitmap;
 } finally {
 if (inputStream != null)
 inputStream.close();
 }
entity.consumeContent();
 }
 } catch (Exception e) {
 // 你可以为IOException提供更明确的错误信息
 getRequest.abort();
Log.e("ImageDownloader", "从" +
 " URL " + url + "检索位图时出错：" + e.toString());
 }

 return null;
}
}

```

```

 "Wait", "Downloading Image");

}

@Override
protected void onPostExecute(Bitmap result) {
 Log.i("Async-Example", "onPostExecute Called");
 downloadedImg.setImageBitmap(result);
simpleWaitDialog.dismiss();
}

private Bitmap downloadBitmap(String url) {
 // initialize the default HTTP client object
 final DefaultHttpClient client = new DefaultHttpClient();

 // forming a HttpGet request
 final HttpGet getRequest = new HttpGet(url);
 try {

HttpResponse response = client.execute(getRequest);

 // check 200 OK for success
 final int statusCode = response.getStatusLine().getStatusCode();

 if (statusCode != HttpStatus.SC_OK) {
Log.w("ImageDownloader", "Error " + statusCode +
 " while retrieving bitmap from " + url);
 return null;
 }

 final HttpEntity entity = response.getEntity();
 if (entity != null) {
 InputStream inputStream = null;
 try {
 // getting contents from the stream
 inputStream = entity.getContent();

 // decoding stream data back into image Bitmap that android understands
 final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

 return bitmap;
 } finally {
 if (inputStream != null)
 inputStream.close();
 }
entity.consumeContent();
 }
 } catch (Exception e) {
 // You Could provide a more explicit error message for IOException
 getRequest.abort();
Log.e("ImageDownloader", "Something went wrong while" +
 " retrieving bitmap from " + url + e.toString());
 }

 return null;
}
}

```

由于目前示例中没有评论字段（或者我没找到，或者我没有权限），这里有一些关于此的评论：

这是一个展示AsyncTask能做什么的好例子。

不过该示例目前存在以下问题

- 可能的内存泄漏
- 如果在异步任务完成前屏幕旋转，应用可能崩溃。

详情请参见：

- 将Activity作为WeakReference传递以避免内存泄漏
- <http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks>
- 避免AsyncTask导致Activity泄漏

## 第250.4节：取消AsyncTask

```
YourAsyncTask 任务 = new YourAsyncTask();
task.execute();
task.cancel();
```

这不会停止正在进行中的任务，它只是设置了取消标志，可以通过检查 `isCancelled()` 的返回值（假设你的代码当前正在运行）来检测，方法如下：

```
class YourAsyncTask extends AsyncTask<Void, Void, Void> {
 @Override
 protected Void doInBackground(Void... params) {
 while(!isCancelled()) {
 ... 执行长时间任务内容
 //执行你需要的操作，例如上传文件的一部分
 if (isCancelled()) {
 return null; // 任务被检测为已取消
 }
 if (yourTaskCompleted) {
 return null;
 }
 }
 }
}
```

### 注意

如果 `AsyncTask` 在 `doInBackground(Params... params)` 仍在执行时被取消，那么 `doInBackground(Params... params)` 返回后，`onPostExecute(Result result)` 将不会被调用。`AsyncTask` 会调用 `onCancelled(Result result)` 来表示任务在执行过程中被取消。

## 第250.5节：AsyncTask：任务的串行执行和并行执行

`AsyncTask` 是一个抽象类，并不继承 `Thread` 类。它有一个 **abstract** 方法 `doInBackground(Params... params)`，该方法被重写以执行任务。此方法由 `AsyncTask.call()` 调用。

Since there is currently no comment field for examples (or I haven't found it or I haven't permission for it) here is some comment about this:

This is a good example what can be done with `AsyncTask`.

However the example currently has problems with

- possible memory leaks
- app crash if there was a screen rotation shortly before the async task finished.

For details see:

- Pass Activity as WeakReference to avoid memory leaks
- <http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks>
- Avoid leaking Activities with `AsyncTask`

## Section 250.4: Canceling AsyncTask

```
YourAsyncTask task = new YourAsyncTask();
task.execute();
task.cancel();
```

This doesn't stop your task if it was in progress, it just sets the cancelled flag which can be checked by checking the return value of `isCancelled()` (assuming your code is currently running) by doing this:

```
class YourAsyncTask extends AsyncTask<Void, Void, Void> {
 @Override
 protected Void doInBackground(Void... params) {
 while(!isCancelled()) {
 ... doing long task stuff
 //Do something, you need, upload part of file, for example
 if (isCancelled()) {
 return null; // Task was detected as canceled
 }
 if (yourTaskCompleted) {
 return null;
 }
 }
 }
}
```

### Note

If an `AsyncTask` is canceled while `doInBackground(Params... params)` is still executing then the method `onPostExecute(Result result)` will **NOT** be called after `doInBackground(Params... params)` returns. The `AsyncTask` will instead call the `onCancelled(Result result)` to indicate that the task was cancelled during execution.

## Section 250.5: AsyncTask: Serial Execution and Parallel Execution of Task

`AsyncTask` is an abstract Class and does not inherit the `Thread` class. It has an **abstract** method `doInBackground(Params... params)`, which is overridden to perform the task. This method is called from `AsyncTask.call()`.

Executor 是 java.util.concurrent 包的一部分。

此外, AsyncTask 包含 2 个 Executor

#### THREAD\_POOL\_EXECUTOR

它使用工作线程来并行执行任务。

```
public static final Executor THREAD_POOL_EXECUTOR = new ThreadPoolExecutor(CORE_POOL_SIZE,
MAXIMUM_POOL_SIZE, KEEP_ALIVE, TimeUnit.SECONDS, sPoolWorkQueue, sThreadFactory);
```

#### SERIAL\_EXECUTOR

它按顺序执行任务, 即一个接一个地执行。

```
private static class SerialExecutor implements Executor { }
```

两个 Executor 都是 **static**, 因此只存在一个 THREAD\_POOL\_EXECUTOR 和一个 SerialExecutor 对象, 但你可以创建多个 AsyncTask 对象。

因此, 如果您尝试使用默认的执行器 (SerialExecutor) 执行多个后台任务, 这些任务将被排队并依次执行。

如果你尝试使用THREAD\_POOL\_EXECUTOR执行多个后台任务, 那么它们将会并行执行。

示例：

```
public class MainActivity extends Activity {
 private Button bt;
 private int CountTask = 0;
 private static final String TAG = "AsyncTaskExample";

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 bt = (Button) findViewById(R.id.button);
 bt.setOnClickListener(new View.OnClickListener() {
 @Override
 public void 点击事件(View v) {
 BackgroundTask backgroundTask = new BackgroundTask ();
 Integer data[] = { ++CountTask, null, null };

 // 任务在线程池中执行 (1)
 backgroundTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, data);

 // 任务串行执行 (2)
 // 取消注释下面的代码并注释上面线程池执行器的代码, 然后进行测试
 // backgroundTask.execute(data);
 Log.d(TAG, "任务 = " + (int) CountTask + " 任务已排队");
 }
 });
 }

 private class BackgroundTask extends AsyncTask<Integer, Integer, Integer> {
 int 任务编号;
```

Executor are part of java.util.concurrent package.

Moreover, AsyncTask contains 2 Executors

#### THREAD\_POOL\_EXECUTOR

It uses worker threads to execute the tasks parallelly.

```
public static final Executor THREAD_POOL_EXECUTOR = new ThreadPoolExecutor(CORE_POOL_SIZE,
MAXIMUM_POOL_SIZE, KEEP_ALIVE, TimeUnit.SECONDS, sPoolWorkQueue, sThreadFactory);
```

#### SERIAL\_EXECUTOR

It executes the task serially, i.e. one by one.

```
private static class SerialExecutor implements Executor { }
```

Both Executors are **static**, hence only one THREAD\_POOL\_EXECUTOR and one SerialExecutor objects exist, but you can create several AsyncTask objects.

Therefore, if you try to do multiple background task with the default Executor (SerialExecutor), these task will be queue and executed serially.

If you try to do multiple background task with THREAD\_POOL\_EXECUTOR, then they will be executed parallelly.

Example:

```
public class MainActivity extends Activity {
 private Button bt;
 private int CountTask = 0;
 private static final String TAG = "AsyncTaskExample";

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 bt = (Button) findViewById(R.id.button);
 bt.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
 BackgroundTask backgroundTask = new BackgroundTask ();
 Integer data[] = { ++CountTask, null, null };

 // Task Executed in thread pool (1)
 backgroundTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, data);

 // Task executed Serially (2)
 // Uncomment the below code and comment the above code of Thread
 // pool Executor and check
 // backgroundTask.execute(data);
 Log.d(TAG, "Task = " + (int) CountTask + " Task Queued");
 }
 });
 }

 private class BackgroundTask extends AsyncTask<Integer, Integer, Integer> {
 int taskNumber;
```

```

@Override
protected Integer doInBackground(Integer... integers) {
 任务编号 = integers[0];

 try {
 Thread.sleep(1000);
 } catch (InterruptedException e) {
 // TODO 自动生成的 catch 块
 }
 e.printStackTrace();
}

Log.d(TAG, "任务 = " + 任务编号 + " 任务在后台运行");

 publishProgress(任务编号);
 return null;
}

@Override
protected void onPreExecute() {
 super.onPreExecute();
}

@Override
protected void onPostExecute(Integer aLong) {
 super.onPostExecute(aLong);
}

@Override
protected void onProgressUpdate(Integer... values) {
 super.onProgressUpdate(values);
}
Log.d(TAG, "任务 = " + (int) values[0]
 + "任务执行完成");
}
}

```

点击按钮多次以启动任务并查看结果。

### 任务在线程池(1)中执行

每个任务需要1000毫秒完成。

在t=36秒时，任务2、3和4被排队并开始执行，因为它们是并行执行的。

```

08-02 19:48:35.815: D/AsyncTaskExample(11693): 任务 = 1 任务已排队
08-02 19:48:35.815: D/AsyncTaskExample(11693): 任务 = 1 任务在后台运行
08-02 19:48:36.025**: D/AsyncTaskExample(11693): 任务 = 2 任务已排队
08-02 19:48:36.025**: D/AsyncTaskExample(11693): 任务 = 2 任务在后台运行
08-02 19:48:36.165**: D/AsyncTaskExample(11693): 任务 = 3 任务已排队
08-02 19:48:36.165**: D/AsyncTaskExample(11693): 任务 = 3 任务在后台运行
08-02 19:48:36.325**: D/AsyncTaskExample(11693): 任务 = 4 任务已排队
08-02 19:48:36.325**: D/AsyncTaskExample(11693): 任务 = 4 任务在后台运行
08-02 19:48:36.815**: D/AsyncTaskExample(11693): 任务 = 1 任务执行完成
08-02 19:48:36.915**: D/AsyncTaskExample(11693): 任务 = 5 任务已排队
08-02 19:48:36.915**: D/AsyncTaskExample(11693): 任务 = 5 任务在后台运行
08-02 19:48:37.025: D/AsyncTaskExample(11693): 任务 = 2 任务执行完成
08-02 19:48:37.165: D/AsyncTaskExample(11693): 任务 = 3 任务执行完成

```

注释 任务在线程池中执行 (1) 并取消注释 任务串行执行 (2)。

```

@Override
protected Integer doInBackground(Integer... integers) {
 taskNumber = integers[0];

 try {
 Thread.sleep(1000);
 } catch (InterruptedException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

 Log.d(TAG, "Task = " + taskNumber + " Task Running in Background");

 publishProgress(taskNumber);
 return null;
}

@Override
protected void onPreExecute() {
 super.onPreExecute();
}

@Override
protected void onPostExecute(Integer aLong) {
 super.onPostExecute(aLong);
}

@Override
protected void onProgressUpdate(Integer... values) {
 super.onProgressUpdate(values);
}
Log.d(TAG, "Task = " + (int) values[0]
 + " Task Execution Completed");
}
}

```

Perform Click on button several times to start a task and see the result.

### Task Executed in thread pool(1)

Each task takes 1000 ms to complete.

At t=36s, tasks 2, 3 and 4 are queued and started executing also because they are executing parallelly.

```

08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Queued
08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Running in Background
08-02 19:48:36.025**: D/AsyncTaskExample(11693): Task = 2 Task Queued
08-02 19:48:36.025**: D/AsyncTaskExample(11693): Task = 2 Task Running in Background
08-02 19:48:36.165**: D/AsyncTaskExample(11693): Task = 3 Task Queued
08-02 19:48:36.165**: D/AsyncTaskExample(11693): Task = 3 Task Running in Background
08-02 19:48:36.325**: D/AsyncTaskExample(11693): Task = 4 Task Queued
08-02 19:48:36.325**: D/AsyncTaskExample(11693): Task = 4 Task Running in Background
08-02 19:48:36.815**: D/AsyncTaskExample(11693): Task = 1 Task Execution Completed
08-02 19:48:36.915**: D/AsyncTaskExample(11693): Task = 5 Task Queued
08-02 19:48:36.915**: D/AsyncTaskExample(11693): Task = 5 Task Running in Background
08-02 19:48:37.025: D/AsyncTaskExample(11693): Task = 2 Task Execution Completed
08-02 19:48:37.165: D/AsyncTaskExample(11693): Task = 3 Task Execution Completed

```

Comment Task Executed in thread pool(1) and uncomment Task executed Serially (2).

点击按钮多次以启动任务并查看结果。

它是串行执行任务，因此每个任务都是在当前任务执行完成后才开始的。因此当任务1的执行完成后，只有任务2开始在后台运行。反之亦然。

```
08-02 19:42:57.505: D/AsyncTaskExample(10299): 任务 = 1 任务已排队
08-02 19:42:57.505: D/AsyncTaskExample(10299): 任务 = 1 后台任务运行中
08-02 19:42:57.675: D/AsyncTaskExample(10299): 任务 = 2 任务排队中
08-02 19:42:57.835: D/AsyncTaskExample(10299): 任务 = 3 任务排队中
08-02 19:42:58.005: D/AsyncTaskExample(10299): 任务 = 4 任务排队中
08-02 19:42:58.155: D/AsyncTaskExample(10299): 任务 = 5 任务排队中
08-02 19:42:58.505: D/AsyncTaskExample(10299): 任务 = 1 任务执行完成
08-02 19:42:58.505: D/AsyncTaskExample(10299): 任务 = 2 后台任务运行中
08-02 19:42:58.755: D/AsyncTaskExample(10299): 任务 = 6 任务排队中
08-02 19:42:59.295: D/AsyncTaskExample(10299): 任务 = 7 任务排队中
08-02 19:42:59.505: D/AsyncTaskExample(10299): 任务 = 2 任务执行完成
08-02 19:42:59.505: D/AsyncTaskExample(10299): 任务 = 3 后台运行中
08-02 19:43:00.035: D/AsyncTaskExample(10299): 任务 = 8 排队中
08-02 19:43:00.505: D/AsyncTaskExample(10299): 任务 = 3 执行完成
08-02 19:43:**00.505**: D/AsyncTaskExample(10299): 任务 = 4 后台运行中
08-02 19:43:**01.505**: D/AsyncTaskExample(10299): 任务 = 4 执行完成
08-02 19:43:**01.515**: D/AsyncTaskExample(10299): 任务 = 5 后台运行中
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): 任务 = 5 执行完成
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): 任务 = 6 后台运行中
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): 任务 = 7 后台运行中
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): 任务 = 6 执行完成
08-02 19:43:04.515: D/AsyncTaskExample(10299): 任务 = 8 任务正在后台运行
08-02 19:43:**04.515**: D/AsyncTaskExample(10299): 任务 = 7 任务执行完成
```

## 第250.6节：执行顺序

最初引入时，AsyncTasks 在单个后台线程上串行执行。从DONUT开始，改为线程池，允许多个任务并行运行。从HONEYCOMB开始，任务在单个线程上执行，以避免因并行执行导致的常见应用错误。

如果您确实想要并行执行，可以调用 `executeOnExecutor(java.util.concurrent.Executor, Object[])` 并传入 `THREAD_POOL_EXECUTOR`。

SERIAL\_EXECUTOR -> 一个按顺序一次执行一个任务的执行器。

THREAD\_POOL\_EXECUTOR -> 一个可用于并行执行任务的执行器。

示例：

```
任务 task = new 任务();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
 task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR, data);
else
 task.execute(data);
```

## 第250.7节：发布进度

有时，我们需要更新由AsyncTask完成的计算进度。该进度可以用字符串、整数等表示。为此，我们必须使用两个函数。首先，我们需要设置`onProgressUpdate`函数，其参数类型与我们的AsyncTask的第二个类型参数相同。

Perform Click on button several times to start a task and see the result.

It is executing the task serially hence every task is started after the current task completed execution. Hence when Task 1's execution completes, only Task 2 starts running in background. Vice versa.

```
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Queued
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Running in Background
08-02 19:42:57.675: D/AsyncTaskExample(10299): Task = 2 Task Queued
08-02 19:42:57.835: D/AsyncTaskExample(10299): Task = 3 Task Queued
08-02 19:42:58.005: D/AsyncTaskExample(10299): Task = 4 Task Queued
08-02 19:42:58.155: D/AsyncTaskExample(10299): Task = 5 Task Queued
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 1 Task Execution Completed
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 2 Task Running in Background
08-02 19:42:58.755: D/AsyncTaskExample(10299): Task = 6 Task Queued
08-02 19:42:59.295: D/AsyncTaskExample(10299): Task = 7 Task Queued
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 2 Task Execution Completed
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 3 Task Running in Background
08-02 19:43:00.035: D/AsyncTaskExample(10299): Task = 8 Task Queued
08-02 19:43:00.505: D/AsyncTaskExample(10299): Task = 3 Task Execution Completed
08-02 19:43:**00.505**: D/AsyncTaskExample(10299): Task = 4 Task Running in Background
08-02 19:43:**01.505**: D/AsyncTaskExample(10299): Task = 4 Task Execution Completed
08-02 19:43:**01.515**: D/AsyncTaskExample(10299): Task = 5 Task Running in Background
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 5 Task Execution Completed
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 6 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 7 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 6 Task Execution Completed
08-02 19:43:04.515: D/AsyncTaskExample(10299): Task = 8 Task Running in Background
08-02 19:43:**04.515**: D/AsyncTaskExample(10299): Task = 7 Task Execution Completed
```

## Section 250.6: Order of execution

When first introduced, AsyncTasks were executed serially on a single background thread. Starting with DONUT, this was changed to a pool of threads allowing multiple tasks to operate in parallel. Starting with HONEYCOMB, tasks are executed on a single thread to avoid common application errors caused by parallel execution.

If you truly want parallel execution, you can invoke `executeOnExecutor(java.util.concurrent.Executor, Object[])` with `THREAD_POOL_EXECUTOR`.

SERIAL\_EXECUTOR -> An Executor that executes tasks one at a time in serial order.

THREAD\_POOL\_EXECUTOR -> An Executor that can be used to execute tasks in parallel.

sample :

```
Task task = new Task();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
 task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR, data);
else
 task.execute(data);
```

## Section 250.7: Publishing progress

Sometimes, we need to update the progress of the computation done by an AsyncTask. This progress could be represented by a string, an integer, etc. To do this, we have to use two functions. First, we need to set the `onProgressUpdate` function whose parameter type is the same as the second type parameter of our AsyncTask.

```
class YourAsyncTask extends AsyncTask<URL, Integer, Long> {
 @Override
 protected void onProgressUpdate(Integer... args) {
 setProgressPercent(args[0])
 }
}
```

其次，我们必须在doInBackground函数中使用publishProgress函数，仅此而已，之前的方法将完成所有工作。

```
protected Long doInBackground(URL... urls) {
 int count = urls.length;
 long totalSize = 0;
 for (int i = 0; i < count; i++) {
 totalSize += Downloader.downloadFile(urls[i]);
 publishProgress((int) ((i / (float) count) * 100));
 }
 return totalSize;
}
```

```
class YourAsyncTask extends AsyncTask<URL, Integer, Long> {
 @Override
 protected void onProgressUpdate(Integer... args) {
 setProgressPercent(args[0])
 }
}
```

Second, we have to use the function publishProgress necessarily on the doInBackground function, and that is all, the previous method will do all the job.

```
protected Long doInBackground(URL... urls) {
 int count = urls.length;
 long totalSize = 0;
 for (int i = 0; i < count; i++) {
 totalSize += Downloader.downloadFile(urls[i]);
 publishProgress((int) ((i / (float) count) * 100));
 }
 return totalSize;
}
```

# 第251章：使用Espresso测试UI

## 第251.1节：Espresso概述

设置Espresso：

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

**ViewMatchers** – 一组实现了 `Matcher<? super View>` 接口的对象。你可以将其中一个或多个传递给 `onView` 方法，以定位当前视图层级中的视图。

**ViewActions** – 一组 `ViewActions`，可以传递给 `ViewInteraction.perform()` 方法（例如，`click()`）。

**ViewAssertions** – 一组 `ViewAssertions`，可以传递给 `ViewInteraction.check()` 方法。大多数情况下，你会使用 `matches` 断言，它使用视图匹配器来断言当前选中视图的状态。

[Google的Espresso速查表](#)

# Chapter 251: Testing UI with Espresso

## Section 251.1: Overall Espresso

**Setup Espresso :**

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

**ViewMatchers** – A collection of objects that implement `Matcher<? super View>` interface. You can pass one or more of these to the `onView` method to locate a view within the current view hierarchy.

**ViewActions** – A collection of `ViewActions` that can be passed to the `ViewInteraction.perform()` method (for example, `click()`).

**ViewAssertions** – A collection of `ViewAssertions` that can be passed the `ViewInteraction.check()` method. Most of the time, you will use the `matches` assertion, which uses a `View matcher` to assert the state of the currently selected view.

[Espresso cheat sheet by google](#)

```
onView(ViewMatcher)
 .perform(ViewAction)
 .check(ViewAssertion);
```

## View Matchers

<b>USER PROPERTIES</b>
withId(...) withText(...) withTagKey(...) withTagValue(...) hasContentDescription(...) withContentDescription(...) withHint(...) withSpinnerText(...) hasLinks() hasEllipsizedText() hasMultilineText()
<b>UI PROPERTIES</b>
isDisplayed() isCompletelyDisplayed() isEnabled() hasFocus() isClickable() isChecked() isNotChecked() withEffectiveVisibility(...) isSelected()
<b>OBJECT MATCHER</b>
allOr(Matcher) anyOr(Matcher) is(...) not(...) endsWith(String) startsWith(String) instanceOf(Class)
<b>SEE ALSO</b>
Preference matchers Cursor matchers Layout matchers

```
onData(ObjectMatcher)
 .DataOptions
 .perform(ViewAction)
 .check(ViewAssertion);
```

## Data Options

inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)

## View Actions

<b>CLICK/PRESS</b>
click() doubleClick() longClick() pressBack() pressIMEActionButton() pressKey([int/EspressoKey]) pressMenuKey() closeSoftKeyboard() openLink()
<b>GESTURES</b>
scrollTo() swipeLeft() swipeRight() swipeUp() swipeDown()
<b>TEXT</b>
clearText() typeText(String) typeTextIntoFocusedView(String) replaceText(String)

## View Assertions

<b>POSITION ASSERTIONS</b>
isLeftOf(Matcher) isRightOf(Matcher) isLeftAlignedWith(Matcher) isRightAlignedWith(Matcher) isAbove(Matcher) isBelow(Matcher)
<b>LAYOUT ASSERTIONS</b>
noEllipsizeText(Matcher) noMultilineButtons() noOverlaps([Matcher])
<b>SEE ALSO</b>
matches(Matcher) doesNotExist() selectedDescendantsMatch(...)
isLeftOf(Matcher) isRightOf(Matcher) isLeftAlignedWith(Matcher) isRightAlignedWith(Matcher) isAbove(Matcher) isBelow(Matcher)

```
intended(IntentMatcher);
```

```
intending(IntentMatcher)
 .respondWith(ActivityResult);
```

## Intent Matchers

<b>INTENT</b>
hasAction(...) hasCategories(...) hasData(...) hasComponent(...) hasExtra(...) hasExtras(Matcher) hasExtraWithKey(...) hasType(...) hasPackage() toPackage(String) hasFlag(int) hasFlags(...) isInternal()
<b>URI</b>
hasHost(...) hasParamWithName(...) hasPath(...) hasParamWithValue(...) hasScheme(...) hasSchemeSpecificPart(...)
<b>BUNDLE</b>
hasEntry(...) hasKey(...) hasValue(...)
<b>COMPONENT NAME</b>
hasClassName(...) hasPackageName(...) hasShortClassName(...) hasMyPackageName()

v2.1.0, 4/21/2015

```
onView(ViewMatcher)
 .perform(ViewAction)
 .check(ViewAssertion);
```

## View Matchers

<b>USER PROPERTIES</b>
withId(...) withText(...) withTagKey(...) withTagValue(...) hasContentDescription(...) withContentDescription(...) withHint(...) withSpinnerText(...) hasLinks() hasEllipsizedText() hasMultilineText()
<b>UI PROPERTIES</b>
isDisplayed() isCompletelyDisplayed() isEnabled() hasFocus() isClickable() isChecked() isNotChecked() withEffectiveVisibility(...) isSelected()
<b>OBJECT MATCHER</b>
allOr(Matcher) anyOr(Matcher) is(...) not(...) endsWith(String) startsWith(String) instanceOf(Class)
<b>SEE ALSO</b>
Preference matchers Cursor matchers Layout matchers

```
onData(ObjectMatcher)
 .DataOptions
 .perform(ViewAction)
 .check(ViewAssertion);
```

## Data Options

inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)

## View Actions

<b>CLICK/PRESS</b>
click() doubleClick() longClick() pressBack() pressIMEActionButton() pressKey([int/EspressoKey]) pressMenuKey() closeSoftKeyboard() openLink()
<b>GESTURES</b>
scrollTo() swipeLeft() swipeRight() swipeUp() swipeDown()
<b>TEXT</b>
clearText() typeText(String) typeTextIntoFocusedView(String) replaceText(String)

## View Assertions

<b>POSITION ASSERTIONS</b>
isLeftOf(Matcher) isRightOf(Matcher) isLeftAlignedWith(Matcher) isRightAlignedWith(Matcher) isAbove(Matcher) isBelow(Matcher)
<b>LAYOUT ASSERTIONS</b>
noEllipsizeText(Matcher) noMultilineButtons() noOverlaps([Matcher])
<b>SEE ALSO</b>
matches(Matcher) doesNotExist() selectedDescendantsMatch(...)
isLeftOf(Matcher) isRightOf(Matcher) isLeftAlignedWith(Matcher) isRightAlignedWith(Matcher) isAbove(Matcher) isBelow(Matcher)

```
intended(IntentMatcher);
```

```
intending(IntentMatcher)
 .respondWith(ActivityResult);
```

## Intent Matchers

<b>INTENT</b>
hasAction(...) hasCategories(...) hasData(...) hasComponent(...) hasExtra(...) hasExtras(Matcher) hasExtraWithKey(...) hasType(...) hasPackage() toPackage(String) hasFlag(int) hasFlags(...) isInternal()
<b>URI</b>
hasHost(...) hasParamWithName(...) hasPath(...) hasParamWithValue(...) hasScheme(...) hasSchemeSpecificPart(...)
<b>BUNDLE</b>
hasEntry(...) hasKey(...) hasValue(...)
<b>COMPONENT NAME</b>
hasClassName(...) hasPackageName(...) hasShortClassName(...) hasMyPackageName()

v2.1.0, 4/21/2015

### 在EditText中输入文本

```
onView(withId(R.id.edt_name)).perform(typeText("XYZ"));
 closeSoftKeyboard();
```

### 执行视图点击操作

```
onView(withId(R.id.btn_id)).perform(click());
```

### 检查视图是否显示

```
onView(withId(R.id.edt_pan_number)).check(ViewAssertions.matches(isDisplayed()));
```

## 第251.2节：Espresso简单UI测试

### UI测试工具

目前主要用于UI测试的两种工具是Appium和Espresso。

Appium	Espresso
黑盒测试	白盒/灰盒测试
所见即所测	可以更改应用的内部工作并为测试做准备，例如，在运行测试之前将一些数据保存到数据库或共享偏好设置中
主要用于集成端到端测试和完整的用户流程	测试屏幕和/或流程的功能
可以抽象化，使编写的测试可以在iOS和Android上执行	仅限Android
支持良好	支持良好
支持通过Selenium Grid在多设备上并行测试	不支持开箱即用的并行测试，存在如Spoon的插件，直到真正的Google支持推出

### 如何将 Espresso 添加到项目中

```
dependencies {
 // 设置此依赖以便使用 Android JUnit 运行器
 androidTestCompile 'com.android.support.test:runner:0.5'
 // 设置此依赖以使用 JUnit 4 规则
 androidTestCompile 'com.android.support.test:rules:0.5'
 // 设置此依赖以构建和运行 Espresso 测试
 androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
 // 设置此依赖以构建和运行 UI Automator 测试
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

注意 如果您使用的是最新的支持库、注解等，您需要从 espresso 中排除旧版本以避免冲突：

```
// 测试支持库存在冲突 (详见
http://stackoverflow.com/questions/29857695)
// 因此目前从这里排除 support-annotations 依赖以避免冲突
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
 exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}
```

### Enter Text In EditText

```
onView(withId(R.id.edt_name)).perform(typeText("XYZ"));
 closeSoftKeyboard();
```

### Perform Click on View

```
onView(withId(R.id.btn_id)).perform(click());
```

### Checking View is Displayed

```
onView(withId(R.id.edt_pan_number)).check(ViewAssertions.matches(isDisplayed()));
```

## Section 251.2: Espresso simple UI test

### UI testing tools

Two main tools that are nowadays mostly used for UI testing are Appium and Espresso.

Appium	Espresso
blackbox test	white/gray box testing
what you see is what you can test	can change inner workings of the app and prepare it for testing, e.g. save some data to database or sharedpreferences before running the test
used mostly for integration end to end tests and entire user flows	testing the functionality of a screen and/or flow
can be abstracted so test written can be executed on iOS and Android	Android Only
well supported	well supported
supports parallel testing on multiple devices with selenium grid	Not out of the box parallel testing, plugins like Spoon exists until true Google support comes out

### How to add espresso to the project

```
dependencies {
 // Set this dependency so you can use Android JUnit Runner
 androidTestCompile 'com.android.support.test:runner:0.5'
 // Set this dependency to use JUnit 4 rules
 androidTestCompile 'com.android.support.test:rules:0.5'
 // Set this dependency to build and run Espresso tests
 androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
 // Set this dependency to build and run UI Automator tests
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

**NOTE** If you are using latest support libraries, annotations etc. you need to exclude the older versions from espresso to avoid collisions:

```
// there is a conflict with the test support library (see
http://stackoverflow.com/questions/29857695)
// so for now we exclude the support-annotations dependency from here to avoid clashes
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
 exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}
```

```

// 因为 <http://stackoverflow.com/questions/29216327>, 这里排除更多模块

// 更具体地说是因为 <https://code.google.com/p/android-test-kit/issues/detail?id=139>
// 否则你会在设备上遇到奇怪的崩溃, 在模拟器上遇到 dex 异常
// Espresso-contrib 用于 DatePicker、RecyclerView、Drawer 操作、无障碍检查、
CountingIdlingResource
androidTestCompile('com.android.support.test.espresso:espresso-contrib:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
exclude group: 'com.android.support', module: 'design'
 exclude module: 'support-annotations'
exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
exclude module: 'support-v7'
}

//由于 https://code.google.com/p/android/issues/detail?id=183454 排除了特定包
androidTestCompile('com.android.support.test.espresso:espresso-intents:2.2.2') {
 排除组: 'com.android.support', 模块: 'support-annotations'
exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test.espresso:espresso-web:2.2.2') {
 排除组: 'com.android.support', 模块: 'support-annotations'
 排除模块: 'support-annotations'
exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test:runner:0.5') {
 排除组: 'com.android.support', 模块: 'support-annotations'
 排除模块: 'support-annotations'
exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test:rules:0.5') {
 排除组: 'com.android.support', 模块: 'support-annotations'
 排除模块: 'support-annotations'
exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
exclude module: 'support-v7'
}

```

除了这些导入外, 还需要在 build.gradle 中添加 android 仪器测试运行器  
android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

## 设备设置

对于非不稳定测试, 建议在设备上设置以下选项 :

- 开发者选项 / 禁用动画 - 减少测试的不稳定性
- 开发者选项 / 保持唤醒 - 如果你有专用测试设备, 这个很有用
- 开发者选项 / 日志缓冲区大小 - 如果你在手机上运行非常大的测试套件, 设置为更大数值
- 辅助功能 / 触摸与按住延迟 - 设置为长, 以避免 espresso 点击问题

```

// exclude a couple of more modules here because of <http://stackoverflow.com/questions/29216327>
and
// more specifically of <https://code.google.com/p/android-test-kit/issues/detail?id=139>
// otherwise you'll receive weird crashes on devices and dex exceptions on emulators
// Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
CountingIdlingResource
androidTestCompile('com.android.support.test.espresso:espresso-contrib:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
exclude group: 'com.android.support', module: 'design'
 exclude module: 'support-annotations'
exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}

//excluded specific packages due to https://code.google.com/p/android/issues/detail?id=183454
androidTestCompile('com.android.support.test.espresso:espresso-intents:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test.espresso:espresso-web:2.2.2') {
 exclude group: 'com.android.support', module: 'support-annotations'
 exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test:runner:0.5') {
 exclude group: 'com.android.support', module: 'support-annotations'
 exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}

androidTestCompile('com.android.support.test:rules:0.5') {
 exclude group: 'com.android.support', module: 'support-annotations'
 exclude module: 'support-annotations'
 exclude module: 'recyclerview-v7'
 exclude module: 'support-v4'
 exclude module: 'support-v7'
}

```

Other than these imports it is necessary to add android instrumentation test runner to build.gradle  
android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

## Device setup

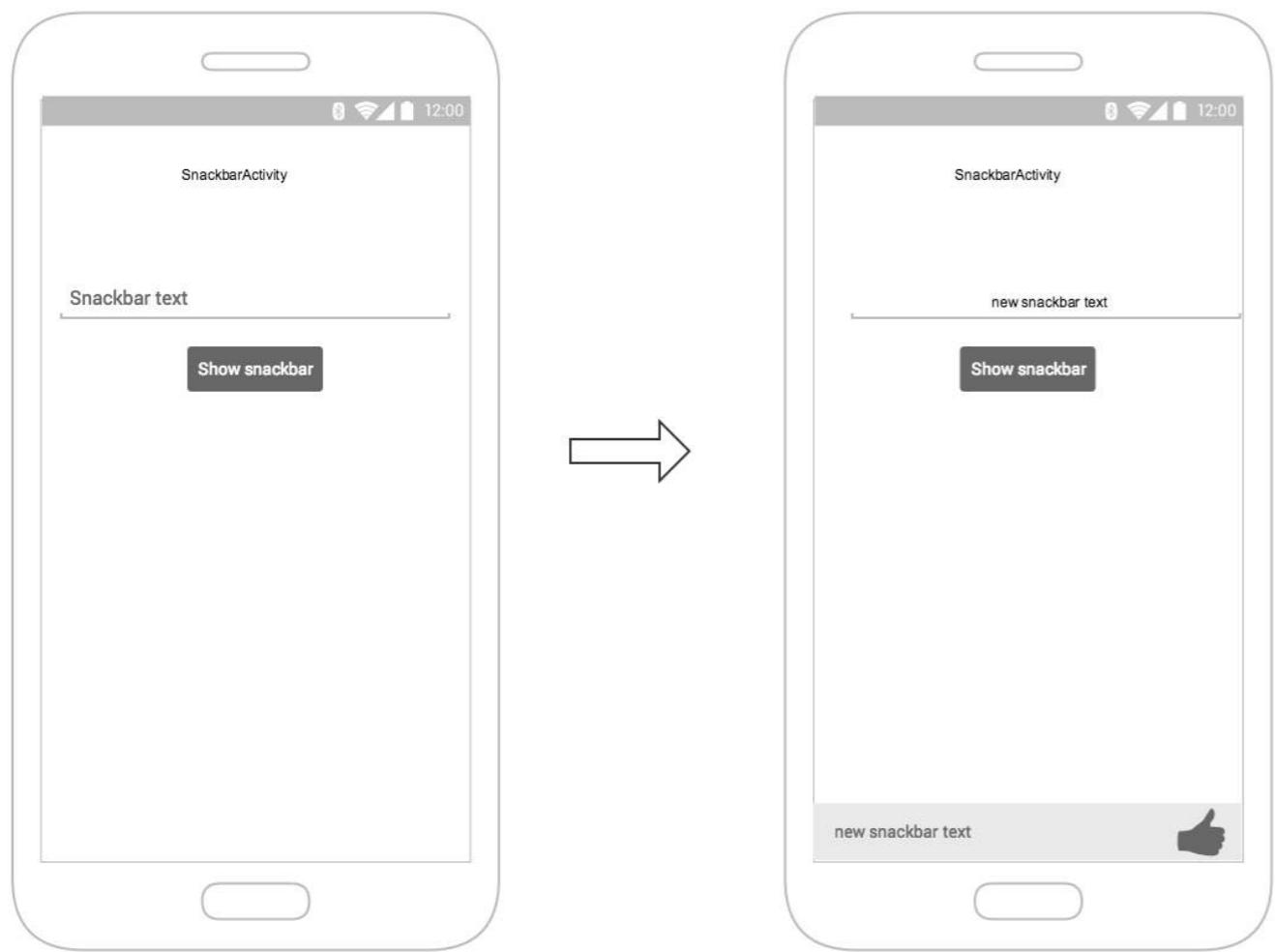
For non flaky test it is recommended to set following settings on your devices:

- Developer options / Disable Animations - reduces flakiness of tests
- Developer options / Stay awake - if you have dedicated devices for tests this is useful
- Developer options / Logger buffer sizes - set to higher number if you run very big test suites on your phone
- Accessibility / Touch & Hold delay - long to avoid problems with tapping in espresso

现实世界中的设置相当复杂吧？好了，既然已经说清楚了，我们来看看如何设置一个小测试。

## 编写测试

假设我们有以下界面：



该界面包含：

- 文本输入框 - R.id.textEntry
- 点击时显示带有输入文本的Snackbar的按钮 - R.id.showSnackbarBtn
- Snackbar应包含用户输入的文本 - android.support.design.R.id.snackbar\_text

现在让我们创建一个类来测试我们的流程：

```
/**
 * Snackbar活动的测试。
 **/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
 // espresso 规则, 指定启动哪个活动

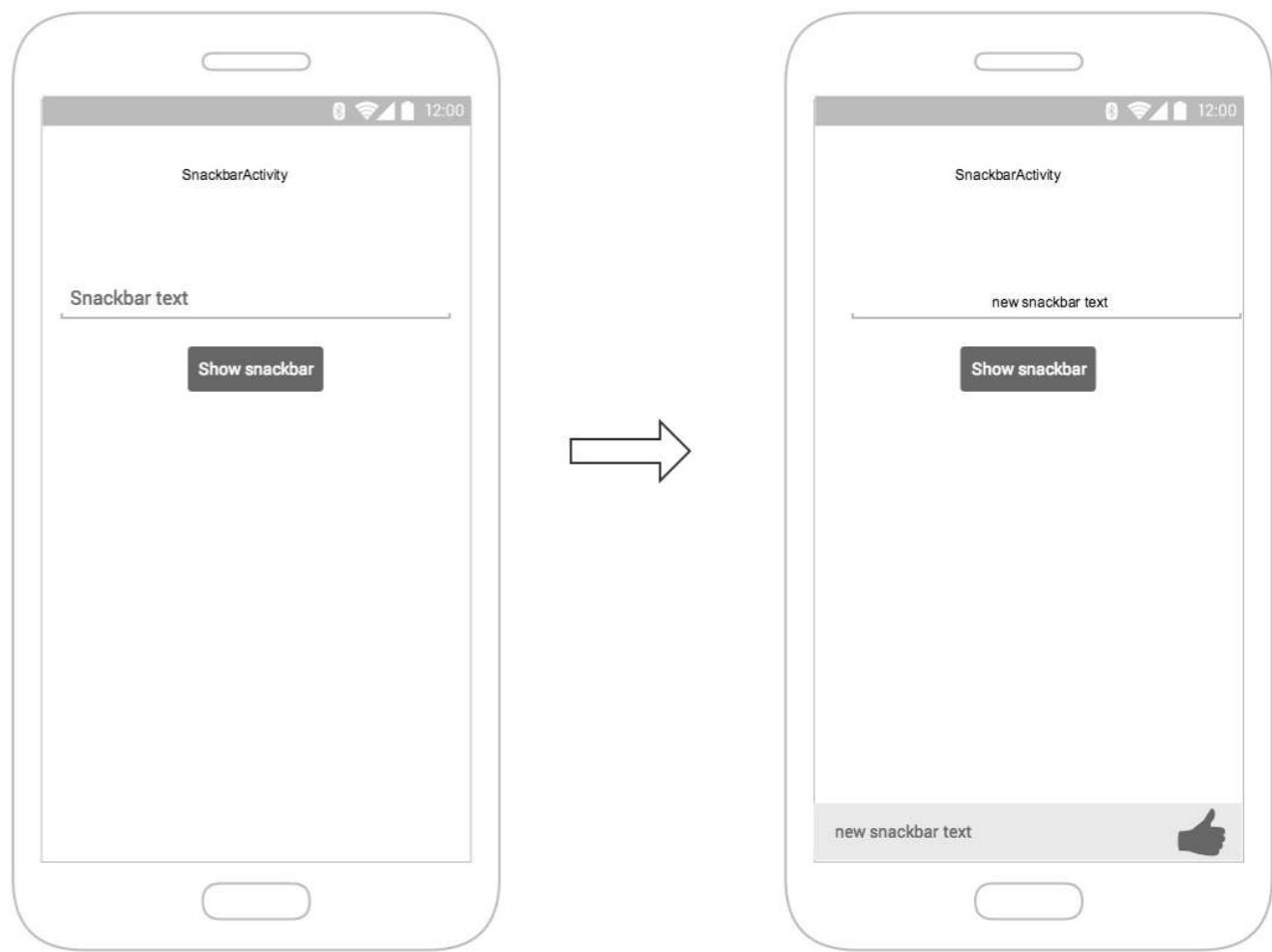
 public final ActivityTestRule<SnackbarActivity> mActivityRule =
 new ActivityTestRule<>(SnackbarActivity.class, true, false);

 @Override
 public void tearDown() throws Exception {
 super.tearDown();
 // 仅作为示例, 说明 tearDown 应该如何清理自身
```

Quite a setup from the real world ha? Well now when that's out of the way let's take a look how to setup a small test

## Writing the test

Let's assume that we have the following screen:



The screen contains:

- text input field - R.id.textEntry
- button which shows Snackbar with typed text when clicked - R.id.showSnackbarBtn
- Snackbar which should contain user typed text - android.support.design.R.id.snackbar\_text

Now let's create a class that will test our flow:

```
/**
 * Testing of the Snackbar activity.
 **/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
 // espresso rule which tells which activity to start
 @Rule
 public final ActivityTestRule<SnackbarActivity> mActivityRule =
 new ActivityTestRule<>(SnackbarActivity.class, true, false);

 @Override
 public void tearDown() throws Exception {
 super.tearDown();
 // just an example how tearDown should cleanup after itself
```

```

mDatabase.clear();
mSharedPrefs.clear();
}

@Override
public void setUp() throws Exception {
super.setUp();
// 设置你的应用，例如如果你需要在 shared
//preferences 中保存用户以保持登录状态，可以在所有测试的 setup 中完成
User mUser = new User();
mUser.setToken("randomToken");
}

/**
* 测试方法应始终以 "testXYZ" 开头，且最好以你想测试的意图命名
*/
@Test
public void testSnackbarIsShown() {
// 启动我们的活动
mActivityRule.launchActivity(null);
// 检查我们的文本输入框是否显示，并输入一些文本
String textToType="new snackbar text";
onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
onView(withId(R.id.textEntry)).perform(typeText(textToType));
// 点击按钮以显示snackbar
onView(withId(R.id.showSnackbarBtn)).perform(click());
// 断言带有snackbar_id且文本为我们输入的文本的视图已显示
onView(allOf(withId(android.support.design.R.id.snackbar_text),
withText(textToType))).check(matches(isDisplayed()));
}
}

```

正如你注意到的，有3-4个你可能经常看到的东西：

**onView(withIdXYZ)** <-- viewMatchers，使用它们可以在屏幕上查找元素

**perform(click())** <-- viewActions，可以对之前找到的元素执行操作

**check(matches(isDisplayed()))** <-- viewAssertions，对之前找到的屏幕元素进行检查

所有这些以及更多内容可以在这里找到：

<https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html>

就是这样，现在你可以通过右键点击类名/测试并选择运行测试，或者使用以下命令来运行测试：

./gradlew connectedFLAVORNAMEAndroidTest

## 第251.3节：打开关闭DrawerLayout

```

public final class DrawerLayoutTest {

@Test public void Open_Close_Drawer_Layout() {
onView(withId(R.id.drawer_layout)).perform(actionOpenDrawer());
onView(withId(R.id.drawer_layout)).perform(actionCloseDrawer());
}

public static ViewAction actionOpenDrawer() {

```

```

mDatabase.clear();
mSharedPrefs.clear();
}

@Override
public void setUp() throws Exception {
super.setUp();
// setting up your application, for example if you need to have a user in shared
//preferences to stay logged in you can do that for all tests in your setup
User mUser = new User();
mUser.setToken("randomToken");
}

/**
* Test methods should always start with "testXYZ" and it is a good idea to
* name them after the intent what you want to test
*/
@Test
public void testSnackbarIsShown() {
// start our activity
mActivityRule.launchActivity(null);
// check is our text entry displayed and enter some text to it
String textToType="new snackbar text";
onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
onView(withId(R.id.textEntry)).perform(typeText(textToType));
// click the button to show the snackbar
onView(withId(R.id.showSnackbarBtn)).perform(click());
// assert that a view with snackbar_id with text which we typed and is displayed
onView(allOf(withId(android.support.design.R.id.snackbar_text),
withText(textToType))).check(matches(isDisplayed()));
}
}

```

As you noticed there are 3-4 things that you might notice come often:

**onView(withIdXYZ)** <-- viewMatchers with them you are able to find elements on screen

**perform(click())** <-- viewActions, you can execute actions on elements you previously found

**check(matches(isDisplayed()))** <-- viewAssertions, checks you want to do on screens you previously found

All of these and many others can be found here:

<https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html>

That's it, now you can run the test either with right clicking on the class name / test and selecting Run test or with command:

./gradlew connectedFLAVORNAMEAndroidTest

## Section 251.3: Open Close DrawerLayout

```

public final class DrawerLayoutTest {

@Test public void Open_Close_Drawer_Layout() {
onView(withId(R.id.drawer_layout)).perform(actionOpenDrawer());
onView(withId(R.id.drawer_layout)).perform(actionCloseDrawer());
}

public static ViewAction actionOpenDrawer() {

```

```

 return new ViewAction() {
@Override public Matcher<View> getConstraints() {
 return assignableFrom(DrawerLayout.class);
}

@Override public String getDescription() {
 return "打开抽屉";
}

@Override public void perform(UiController uiController, View view) {
 ((DrawerLayout) view).openDrawer(GravityCompat.START);
}
};

public static ViewAction actionCloseDrawer() {
 return new ViewAction() {
@Override public Matcher<View> getConstraints() {
 return assignableFrom(DrawerLayout.class);
}

@Override public String getDescription() {
 return "关闭抽屉";
}

@Override public void perform(UiController uiController, View view) {
 ((DrawerLayout) view).closeDrawer(GravityCompat.START);
}
};
}

```

```

 return new ViewAction() {
@Override public Matcher<View> getConstraints() {
 return assignableFrom(DrawerLayout.class);
}

@Override public String getDescription() {
 return "open drawer";
}

@Override public void perform(UiController uiController, View view) {
 ((DrawerLayout) view).openDrawer(GravityCompat.START);
}
};

public static ViewAction actionCloseDrawer() {
 return new ViewAction() {
@Override public Matcher<View> getConstraints() {
 return assignableFrom(DrawerLayout.class);
}

@Override public String getDescription() {
 return "close drawer";
}

@Override public void perform(UiController uiController, View view) {
 ((DrawerLayout) view).closeDrawer(GravityCompat.START);
}
};
}

```

## 第251.4节：设置Espresso

在您的Android应用模块的build.gradle文件中添加以下依赖项：

```

dependencies {
 // Android JUnit 运行器
 androidTestCompile 'com.android.support.test:runner:0.5'
 // JUnit4 规则
 androidTestCompile 'com.android.support.test:rules:0.5'
 // Espresso 核心
 androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
 // Espresso-contrib 用于 DatePicker、RecyclerView、Drawer 操作、无障碍检查、
 CountingIdlingResource
 androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2.2'
 // UI Automator 测试
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}

```

在 build.gradle 文件中为 testInstrumentationRunner 参数指定 AndroidJUnitRunner。

```

android {

defaultConfig {
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
}

```

## Section 251.4: Set Up Espresso

In the build.gradle file of your Android app module add next dependencies:

```

dependencies {
 // Android JUnit Runner
 androidTestCompile 'com.android.support.test:runner:0.5'
 // JUnit4 Rules
 androidTestCompile 'com.android.support.test:rules:0.5'
 // Espresso core
 androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
 // Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
 CountingIdlingResource
 androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2.2'
 // UI Automator tests
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}

```

Specify the AndroidJUnitRunner for the testInstrumentationRunner parameter in the build.gradle file.

```

android {

defaultConfig {
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
}

```

此外，添加此依赖以提供意图模拟支持

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

并添加此项以支持 WebView 测试

```
// Espresso-web 用于 WebView 支持
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

## 第 251.5 节：对视图执行操作

可以使用 `perform` 方法对视图执行 `ViewActions` 操作。

`ViewActions` 类提供了最常用操作的辅助方法，例如：

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

例如，点击视图：

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

您可以通过一次 `perform` 调用执行多个操作：

```
onView(...).perform(typeText("Hello"), click());
```

如果您正在操作的视图位于 `ScrollView`（垂直或水平）内，建议在需要视图显示的操作（如 `click()` 和 `typeText()`）之前使用 `scrollTo()`。这确保视图在执行其他操作之前已显示：

```
onView(...).perform(scrollTo(), click());
```

## 第251.6节：使用onView查找视图

使用 `ViewMatchers`，您可以在当前视图层次结构中查找视图。

要查找视图，请使用带有选择正确视图的视图匹配器的 `onView()` 方法。`onView()` 方法返回一个类型为 `ViewInteraction` 的对象。

例如，通过其 `R.id` 查找视图非常简单：

```
onView(withId(R.id.my_view))
```

通过文本查找视图：

```
onView(withText("Hello World"))
```

## 第251.7节：创建Espresso测试类

将以下 Java 类放置在 `src/androidTest/java` 目录下并运行。

```
public class UITest {
```

Additionally, add this dependency for providing intent mocking support

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

And add this one for webview testing support

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

## Section 251.5: Performing an action on a view

It is possible to perform `ViewActions` on a view using the `perform` method.

The `ViewActions` class provides helper methods for the most common actions, like:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

For example, to click on the view:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

You can execute more than one action with one `perform` call:

```
onView(...).perform(typeText("Hello"), click());
```

If the view you are working with is located inside a `ScrollView` (vertical or horizontal), consider preceding actions that require the view to be displayed (like `click()` and `typeText()`) with `scrollTo()`. This ensures that the view is displayed before proceeding to the other action:

```
onView(...).perform(scrollTo(), click());
```

## Section 251.6: Finding a view with onView

With the `ViewMatchers` you can find view in the current view hierarchy.

To find a view, use the `onView()` method with a view matcher which selects the correct view. The `onView()` methods return an object of type `ViewInteraction`.

For example, finding a view by its `R.id` is as simple as:

```
onView(withId(R.id.my_view))
```

Finding a view with a text:

```
onView(withText("Hello World"))
```

## Section 251.7: Create Espresso Test Class

Place next java class in `src/androidTest/java` and run it.

```
public class UITest {
```

```
@Test public void Simple_Test() {
 onView(withId(R.id.my_view)) // withId(R.id.my_view) 是一个ViewMatcher
 .perform(click()) // click() 是一个ViewAction
 .check(matches(isDisplayed())); // matches(isDisplayed()) 是一个ViewAssertion
}
}
```

## 第251.8节：向上导航

```
@Test
public void setUpNavigation() {
 intending(hasComponent(ParentActivity.class.getName())).respondWith(new
 Instrumentation.ActivityResult(0, null));

 onView(withContentDescription("向上导航")).perform(click());

 intended(hasComponent(ParentActivity.class.getName()));
}
```

请注意，这是一种变通方法，可能会与其他具有相同内容描述的视图发生冲突。

## 第251.9节：在测试套件中分组一组测试类

您可以通过定义一个Suite来组织您的仪器化单元测试的执行。

```
/**
 * 运行所有单元测试。
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({MyTest1.class ,
 MyTest2.class,
 MyTest3.class})
public class AndroidTestSuite {}
```

然后在AndroidStudio中，您可以通过gradle运行或设置一个新的配置，如下所示：

```
@Test public void Simple_Test() {
 onView(withId(R.id.my_view)) // withId(R.id.my_view) 是一个ViewMatcher
 .perform(click()) // click() 是一个ViewAction
 .check(matches(isDisplayed())); // matches(isDisplayed()) 是一个ViewAssertion
}
}
```

## Section 251.8: Up Navigation

```
@Test
public void setUpNavigation() {
 intending(hasComponent(ParentActivity.class.getName())).respondWith(new
 Instrumentation.ActivityResult(0, null));

 onView(withContentDescription("Navigate up")).perform(click());

 intended(hasComponent(ParentActivity.class.getName()));
}
```

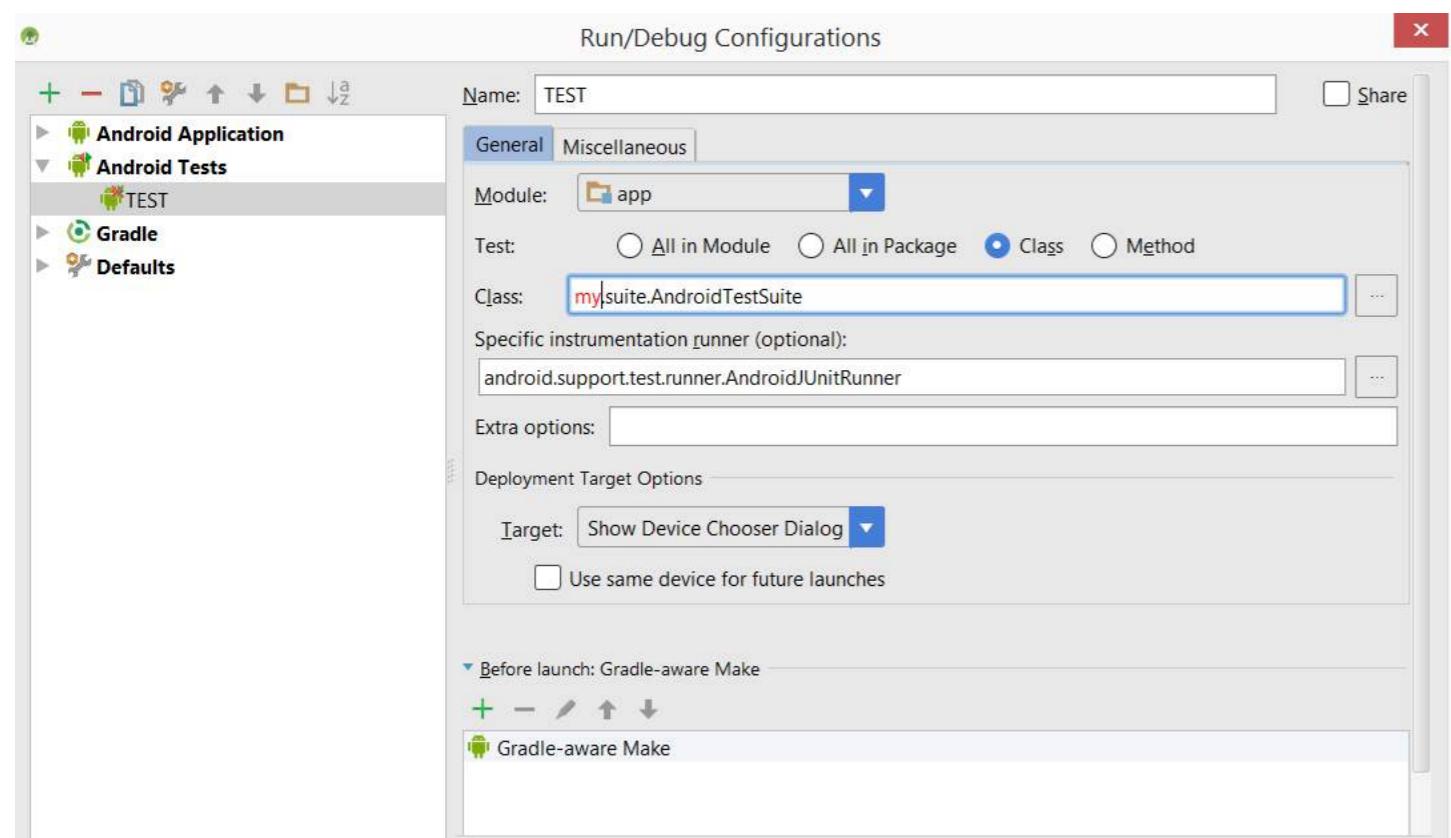
Note that this is a workaround and will collide with other Views that have the same content description.

## Section 251.9: Group a collection of test classes in a test suite

You can organize the execution of your instrumented unit tests defining a [Suite](#).

```
/**
 * Runs all unit tests.
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({MyTest1.class ,
 MyTest2.class,
 MyTest3.class})
public class AndroidTestSuite {}
```

Then in AndroidStudio you can run with gradle or setting a new configuration like:



测试套件可以嵌套。

## 第251.10节：Espresso自定义匹配器

Espresso默认提供了许多匹配器，帮助你找到需要进行检查或交互的视图。

最重要的匹配器可以在以下备忘单中找到：

<https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/>

一些匹配器示例包括：

- `withId(R.id.你要查找的对象的ID);`
- `withText("你期望对象拥有的文本");`
- `isDisplayed() <-- 检查视图是否可见`
- `doesNotExist() <-- 检查视图是否存在`

这些匹配器在日常使用中非常有用，但如果你有更复杂的视图，编写自定义匹配器可以使测试更易读，并且可以在不同地方复用。

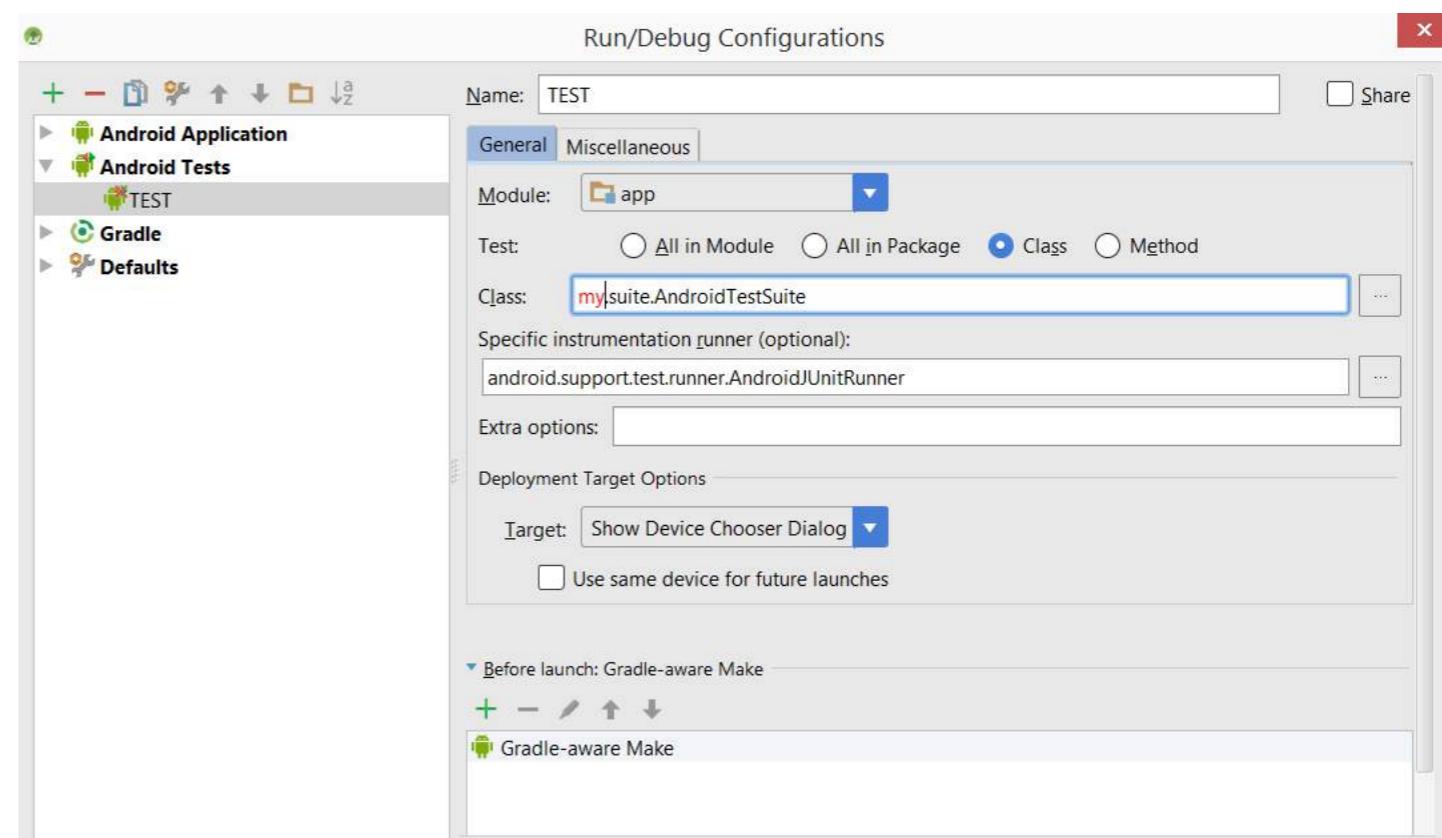
你可以扩展的两种最常见的匹配器类型是：`TypeSafeMatcher` 和 `BoundedMatcher` 实现 `TypeSafeMatcher`

`her` 需要你检查断言的视图实例类型，如果是正确的类型，则将其某些属性与提供给匹配器的值进行匹配。

例如，一个验证 `ImageView` 是否具有正确 `drawable` 的类型安全匹配器：

```
public class DrawableMatcher extends TypeSafeMatcher<View> {

 private @DrawableRes final int expectedId;
 String resourceName;
```



Test suites can be nested.

## Section 251.10: Espresso custom matchers

Espresso by default has many matchers that help you find views that you need to do some checks or interactions with them.

Most important ones can be found in the following cheat sheet:

<https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/>

Some examples of matchers are:

- `withId(R.id.ID_of_object_you_are_looking_for);`
- `withText("Some text you expect object to have");`
- `isDisplayed() <-- check is the view visible`
- `doesNotExist() <-- check that the view does not exist`

All of these are very useful for everyday use, but if you have more complex views writing your custom matchers can make the tests more readable and you can reuse them in different places.

There are 2 most common type of matchers you can extend: **TypeSafeMatcher** **BoundedMatcher**

Implementing `TypeSafeMatcher` requires you to check the `instanceOf` the view you are asserting against, if its the correct type you match some of its properties against a value you provided to a matcher.

For example, type safe matcher that validates an image view has correct drawable:

```
public class DrawableMatcher extends TypeSafeMatcher<View> {

 private @DrawableRes final int expectedId;
 String resourceName;
```

```

public DrawableMatcher(@DrawableRes int expectedId) {
 super(View.class);
 this.expectedId = expectedId;
}

@Override
protected boolean matchesSafely(View target) {
 //Type check we need to do in TypeSafeMatcher
 if (!(target instanceof ImageView)) {
 return false;
 }
 //我们从聚焦的视图中获取 ImageView
 ImageView imageView = (ImageView) target;
 if (expectedId < 0) {
 return imageView.getDrawable() == null;
 }
 //我们从资源中获取要与 ImageView 源进行比较的 drawable
 Resources resources = target.getContext().getResources();
 Drawable expectedDrawable = resources.getDrawable(expectedId);
 resourceName = resources.getResourceEntryName(expectedId);

 if (expectedDrawable == null) {
 return false;
 }
 //比较位图应该给出匹配器是否相等的结果
 Bitmap bitmap = ((BitmapDrawable) imageView.getDrawable()).getBitmap();
 Bitmap otherBitmap = ((BitmapDrawable) expectedDrawable).getBitmap();
 return bitmap.与(otherBitmap)相同;
}

```

```

@Override
public void 描述到(Description description) {
 description.追加文本("来自资源ID的可绘制对象: ");
 description.追加值(expectedId);
 if (resourceName != null) {
 description.追加文本("[");
 description.追加文本(resourceName);
 description.追加文本("]");
 }
}

```

匹配器的使用可以这样封装：

```

public static Matcher<View> 带有可绘制对象(final int resourceId) {
 return new DrawableMatcher(resourceId);
}

onView(带有可绘制对象(R.drawable.someDrawable)).检查(匹配(显示中()));

```

有界匹配器类似，只是你不必做类型检查，因为这会自动为你完成：

```

/**
* 匹配一个 {@link TextInputFormView} 的输入提示与给定的资源ID
*
* @param resourceId
* @return
*/

```

```

public DrawableMatcher(@DrawableRes int expectedId) {
 super(View.class);
 this.expectedId = expectedId;
}

@Override
protected boolean matchesSafely(View target) {
 //Type check we need to do in TypeSafeMatcher
 if (!(target instanceof ImageView)) {
 return false;
 }
 //We fetch the image view from the focused view
 ImageView imageView = (ImageView) target;
 if (expectedId < 0) {
 return imageView.getDrawable() == null;
 }
 //We get the drawable from the resources that we are going to compare with image view source
 Resources resources = target.getContext().getResources();
 Drawable expectedDrawable = resources.getDrawable(expectedId);
 resourceName = resources.getResourceEntryName(expectedId);

 if (expectedDrawable == null) {
 return false;
 }
 //comparing the bitmaps should give results of the matcher if they are equal
 Bitmap bitmap = ((BitmapDrawable) imageView.getDrawable()).getBitmap();
 Bitmap otherBitmap = ((BitmapDrawable) expectedDrawable).getBitmap();
 return bitmap.sameAs(otherBitmap);
}

```

```

@Override
public void 描述到(Description description) {
 description.appendText("with drawable from resource id: ");
 description.appendValue(expectedId);
 if (resourceName != null) {
 description.appendText("[");
 description.appendText(resourceName);
 description.appendText("]");
 }
}

```

Usage of the matcher could be wrapped like this:

```

public static Matcher<View> withDrawable(final int resourceId) {
 return new DrawableMatcher(resourceId);
}

onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));

```

Bounded matchers are similar you just don't have to do the type check but, since that is done automagically for you:

```

/**
* Matches a {@link TextInputFormView}'s input hint with the given resource ID
*
* @param resourceId
* @return
*/

```

```

public static Matcher<View> withTextInputHint(@StringRes final int stringId) {
 return new BoundedMatcher<View, TextInputFormView>(TextInputFormView.class) {
 private String mResourceName = null;

 @Override
 public void describeTo(final Description description) {
 //fill these out properly so your logging and error reporting is more clear
 description.appendText("带有提示的 TextInputFormView ");
 description.appendValue(stringId);
 if (null != mResourceName) {
 description.appendText("[");
 description.appendText(mResourceName);
 description.appendText("]");
 }
 }

 @Override
 public boolean matchesSafely(final TextInputFormView view) {
 if (null == mResourceName) {
 try {
 mResourceName = view.getResources().getResourceEntryName(stringId);
 } catch (Resources.NotFoundException e) {
 throw new IllegalStateException("无法找到ID为 " + stringId 的字符串",
 e);
 }
 }
 return view.getResources().getString(stringId).equals(view.getHint());
 }
 };
}

```

关于匹配器的更多内容可以阅读：

<http://hamcrest.org/>

<https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html>

```

public static Matcher<View> withTextInputHint(@StringRes final int stringId) {
 return new BoundedMatcher<View, TextInputFormView>(TextInputFormView.class) {
 private String mResourceName = null;

 @Override
 public void describeTo(final Description description) {
 //fill these out properly so your logging and error reporting is more clear
 description.appendText("with TextInputFormView that has hint ");
 description.appendValue(stringId);
 if (null != mResourceName) {
 description.appendText("[");
 description.appendText(mResourceName);
 description.appendText("]");
 }
 }

 @Override
 public boolean matchesSafely(final TextInputFormView view) {
 if (null == mResourceName) {
 try {
 mResourceName = view.getResources().getResourceEntryName(stringId);
 } catch (Resources.NotFoundException e) {
 throw new IllegalStateException("could not find string with ID " + stringId,
 e);
 }
 }
 return view.getResources().getString(stringId).equals(view.getHint());
 }
 };
}

```

More on matchers can be read up on:

<http://hamcrest.org/>

<https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html>

# 第252章：编写UI测试 - 安卓

本文档的重点是展示如何编写安卓UI和集成测试的目标和方法。Espresso和UIAutomator由谷歌提供，因此重点应放在这些工具及其各自的封装工具上，例如Appium、Spoon等。

## 第252.1节：MockWebServer示例

如果你的活动（Activity）、片段（Fragment）和UI需要一些后台处理，使用MockWebServer是一个不错的选择，它在安卓设备上本地运行，为你的UI提供一个封闭且可测试的环境。

<https://github.com/square/okhttp/tree/master/mockwebserver>

第一步是添加gradle依赖：

```
testCompile 'com.squareup.okhttp3:mockwebserver:(插入最新版本)'
```

现在运行和使用mock服务器的步骤是：

- 创建mock服务器对象
- 在特定地址和端口启动它（通常是localhost:端口号）
- 为特定请求排队响应
- 开始测试

这在mockwebserver的github页面中有很好的解释，但在我们的情况下，我们希望有一个更好且可重用的方案，适用于所有测试，JUnit规则将在这里很好地发挥作用：

```
/**
 * JUnit 规则，用于启动和停止测试运行器的模拟Web服务器
 */
public class MockServerRule extends UiThreadTestRule {

 private MockWebServer mServer;

 public static final int MOCK_WEBSERVER_PORT = 8000;

 @Override
 public Statement apply(final Statement base, Description description) {
 return new Statement() {
 @Override
 public void evaluate() throws Throwable {
 startServer();
 try {
 base.evaluate();
 } finally {
 stopServer();
 }
 }
 };
 }

 /**
 * 返回已启动的Web服务器实例
 *
 * @return 模拟服务器
 */
 public MockWebServer 服务器() {
```

# Chapter 252: Writing UI tests - Android

Focus of this document is to represent goals and ways how to write android UI and integration tests. Espresso and UIAutomator are provided by Google so focus should be around these tools and their respective wrappers e.g. Appium, Spoon etc.

## Section 252.1: MockWebServer example

In case your activities, fragments and UI require some background processing a good thing to use is a MockWebServer which runs locally on an android device which brings a closed and testable environment for your UI.

<https://github.com/square/okhttp/tree/master/mockwebserver>

First step is including the gradle dependency:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Now steps for running and using the mock server are:

- create mock server object
- start it at specific address and port (usually localhost:portnumber)
- enqueue responses for specific requests
- start the test

This is nicely explained in the github page of the mockwebserver but in our case we want something nicer and reusable for all tests, and JUnit rules will come nicely into play here:

```
/**
 * JUnit rule that starts and stops a mock web server for test runner
 */
public class MockServerRule extends UiThreadTestRule {

 private MockWebServer mServer;

 public static final int MOCK_WEBSERVER_PORT = 8000;

 @Override
 public Statement apply(final Statement base, Description description) {
 return new Statement() {
 @Override
 public void evaluate() throws Throwable {
 startServer();
 try {
 base.evaluate();
 } finally {
 stopServer();
 }
 }
 };
 }

 /**
 * Returns the started web server instance
 *
 * @return mock server
 */
 public MockWebServer server() {
```

```

 return mServer;
 }

 public void 启动服务器() throws IOException, NoSuchAlgorithmException {
 mServer = new MockWebServer();
 try {
 mServer(MOCK_WEBSERVER_PORT);
 } catch (IOException e) {
 throw new IllegalStateException(e,"mock 服务器启动问题");
 }
 }

 public void 停止服务器() {
 try {
 mServer.shutdown();
 } catch (IOException e) {
 Timber.e(e, "mock 服务器关闭错误");
 }
 }
}

```

现在假设我们有与前一个示例完全相同的活动，只是在这种情况下，当我们按下按钮时，应用程序会从网络获取一些内容，例如：<https://someapi.com/name>

这将返回一些文本字符串，该字符串将与Snackbar文本连接，例如：NAME + 你输入的文本。

```

/**
 * 测试带有网络功能的Snackbar活动。
 */
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
 // espresso 规则，指定启动哪个活动

 public final ActivityTestRule<SnackbarActivity> mActivityRule =
 new ActivityTestRule<>(SnackbarActivity.class, true, false);

 //启动模拟网络服务器
 @Rule
 public final MockServerRule mMockServerRule = new MockServerRule();

 @Override
 public void tearDown() throws Exception {
 //与前一个示例相同
 }

 @Override
 public void setUp() throws Exception {
 //与前一个示例相同

 //重要： 将你的应用程序指向你的mockwebserver端点，例如
 AppConfig.setEndpointURL("http://localhost:8000");
 }

 /**
 * 测试方法应始终以 "testXYZ" 开头，且最好以你想测试的意图命名
 */
 @Test
 public void testSnackbarIsShown() {
 //设置模拟网络服务器
 }
}

```

```

 return mServer;
 }

 public void startServer() throws IOException, NoSuchAlgorithmException {
 mServer = new MockWebServer();
 try {
 mServer(MOCK_WEBSERVER_PORT);
 } catch (IOException e) {
 throw new IllegalStateException(e,"mock server start issue");
 }
 }

 public void stopServer() {
 try {
 mServer.shutdown();
 } catch (IOException e) {
 Timber.e(e, "mock server shutdown error");
 }
 }
}

```

Now lets assume that we have the exact same activity like in previous example, just in this case when we push the button app will fetch something from the network for example: <https://someapi.com/name>

This would return some text string which would be concatenated in the snackbar text e.g. NAME + text you typed in.

```

/**
 * Testing of the snackbar activity with networking.
 */
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
 //espresso rule which tells which activity to start
 @Rule
 public final ActivityTestRule<SnackbarActivity> mActivityRule =
 new ActivityTestRule<>(SnackbarActivity.class, true, false);

 //start mock web server
 @Rule
 public final MockServerRule mMockServerRule = new MockServerRule();

 @Override
 public void tearDown() throws Exception {
 //same as previous example
 }

 @Override
 public void setUp() throws Exception {
 //same as previous example

 //IMPORTANT: point your application to your mockwebserver endpoint e.g.
 AppConfig.setEndpointURL("http://localhost:8000");
 }

 /**
 * Test methods should always start with "testXYZ" and it is a good idea to
 * name them after the intent what you want to test
 */
 @Test
 public void testSnackbarIsShown() {
 //setup mockweb server
 }
}

```

```

mMockServerRule.server().setDispatcher(getDispatcher());

mActivityRule.launchActivity(null);
//检查我们的文本输入框是否显示，并输入一些文本
String textToType="new snackbar text";
onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
//我们检查我们的snackbar是否显示来自mock webserver的文本加上我们输入的文本
onView(withId(R.id.textEntry)).perform(typeText("JazzJackTheRabbit" + textToType));
//点击按钮以显示snackbar
onView(withId(R.id.showSnackbarBtn)).perform(click());
//断言带有snackbar_id且文本为我们输入的文本的视图已显示
onView(allOf(withId(android.support.design.R.id.snackbar_text),
 withText(textToType))).check(matches(isDisplayed()));

}

/**
*创建一个带有预录请求和响应的mock web服务器分发器
*/
private Dispatcher getDispatcher() {
 final Dispatcher dispatcher = new Dispatcher() {
 @Override
 public MockResponse dispatch(RecordedRequest request) throws InterruptedException {
 if (request.getPath().equals("/name")){
 return new MockResponse().setResponseCode(200)
 .setBody("JazzJackTheRabbit");
 }
 throw new IllegalStateException("no mock set up for " + request.getPath());
 }
 };
 return dispatcher;
}

```

我建议将调度器包装在某种构建器中，这样你可以轻松地链式添加新的响应用于你的界面。例如：

```

return new DispatcherBuilder()
.withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
 .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
.withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());

```

## 第252.2节：空闲资源 (IdlingResource)

空闲资源的优势在于不必通过 sleep() 等待某个应用的处理（网络、计算、动画等）完成，这会导致测试不稳定和/或延长测试运行时间。官方文档可以在这里找到。

### 实现

实现 IdlingResource 接口时，你需要做三件事：

- `getName()` - 返回您的空闲资源的名称。
- `isIdleNow()` - 检查你的 xyz 对象、操作等当前是否处于空闲状态。
- `registerIdleTransitionCallback (IdlingResource.ResourceCallback callback)` - 提供一个回调，当你的对象转换为空闲状态时，你应该调用该回调。

现在你应该创建自己的逻辑，确定你的应用何时空闲，何时不空闲，因为这取决于应用。下面你会看到一个简单的示例，仅用于展示其工作原理。网上还有其他示例，但具体的应用实现会带来具体的空闲资源实现。

```

mMockServerRule.server().setDispatcher(getDispatcher());

mActivityRule.launchActivity(null);
//check is our text entry displayed and enter some text to it
String textToType="new snackbar text";
onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
//we check is our snackbar showing text from mock webserver plus the one we typed
onView(withId(R.id.textEntry)).perform(typeText("JazzJackTheRabbit" + textToType));
//click the button to show the snackbar
onView(withId(R.id.showSnackbarBtn)).perform(click());
//assert that a view with snackbar_id with text which we typed and is displayed
onView(allOf(withId(android.support.design.R.id.snackbar_text),
 withText(textToType))).check(matches(isDisplayed()));

}

/**
*creates a mock web server dispatcher with prerecorded requests and responses
*/
private Dispatcher getDispatcher() {
 final Dispatcher dispatcher = new Dispatcher() {
 @Override
 public MockResponse dispatch(RecordedRequest request) throws InterruptedException {
 if (request.getPath().equals("/name")){
 return new MockResponse().setResponseCode(200)
 .setBody("JazzJackTheRabbit");
 }
 throw new IllegalStateException("no mock set up for " + request.getPath());
 }
 };
 return dispatcher;
}

```

I would suggest wrapping the dispatcher in some sort of a Builder so you can easily chain and add new responses for your screens. e.g.

```

return new DispatcherBuilder()
 .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
 .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
 .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());

```

## Section 252.2: IdlingResource

The power of idling resources lies in not having to wait for some app's processing (networking, calculations, animations, etc.) to finish with `sleep()`, which brings flakiness and/or prolongs the tests run. The official documentation can be found [here](#).

### Implementation

There are three things that you need to do when implementing IdlingResource interface:

- `getName()` - Returns the name of your idling resource.
- `isIdleNow()` - Checks whether your xyz object, operation, etc. is idle at the moment.
- `registerIdleTransitionCallback (IdlingResource.ResourceCallback callback)` - Provides a callback which you should call when your object transitions to idle.

Now you should create your own logic and determine when your app is idle and when not, since this is dependant on the app. Below you will find a simple example, just to show how it works. There are other examples online, but specific app implementation brings to specific idling resource implementations.

## 注意事项

- 谷歌曾有一些示例将IdlingResources放在应用代码中。请不要这样做。他们大概只是为了展示其工作原理才这么做的。
- 保持代码整洁并遵守单一职责原则是你的责任！

## 示例

假设你有一个活动执行一些奇怪的操作，且片段加载时间很长，导致你的 Espresso 测试失败，因为无法找到片段中的资源（你应该修改活动的创建方式和时间以加快速度）。但无论如何，为了简单起见，以下示例展示了应该是什么样子。

我们的示例空闲资源将获取两个对象：

- 你需要找到的片段的标签，并等待其附加到活动中。
- 用于查找碎片的FragmentManager对象。

```
/**
 * FragmentIdlingResource - 等待碎片加载完成的空闲资源。
 */
public class FragmentIdlingResource implements IdlingResource {
 private final FragmentManager mFragmentManager;
 private final String mTag;
 // 当活动转换为空闲时使用的资源回调
 private volatile ResourceCallback resourceCallback;

 public FragmentIdlingResource(FragmentManager fragmentManager, String tag) {
 mFragmentManager = fragmentManager;
 mTag = tag;
 }

 @Override
 public String getName() {
 return FragmentIdlingResource.class.getName() + ":" + mTag;
 }

 @Override
 public boolean isIdleNow() {
 // 简单检查，如果碎片已添加，则应用已变为空闲
 boolean idle = (mFragmentManager.findFragmentByTag(mTag) != null);
 if (idle) {
 //重要提示：确保调用 onTransitionToIdle
 resourceCallback.onTransitionToIdle();
 }
 return idle;
 }

 @Override
 public void registerIdleTransitionCallback(ResourceCallback resourceCallback) {
 this.resourceCallback = resourceCallback;
 }
}
```

既然你已经编写了IdlingResource，那么你需要在某处使用它，对吧？

## 用法

我们跳过整个测试类的设置，直接看看一个测试用例的样子：

## NOTES

- There have been some Google examples where they put IdlingResources in the code of the app. **Do not do this.** They presumably placed it there just to show how they work.
- Keeping your code clean and maintaining single principle of responsibility is up to you!

## Example

Let us say that you have an activity which does weird stuff and takes a long time for the fragment to load and thus making your Espresso tests fail by not being able to find resources from your fragment (you should change how your activity is created and when to speed it up). But in any case to keep it simple, the following example shows how it should look like.

Our example idling resource would get two objects:

- The **tag** of the fragment which you need to find and waiting to get attached to the activity.
- A **FragmentManager** object which is used for finding the fragment.

```
/**
 * FragmentIdlingResource - idling resource which waits while Fragment has not been loaded.
 */
public class FragmentIdlingResource implements IdlingResource {
 private final FragmentManager mFragmentManager;
 private final String mTag;
 //resource callback you use when your activity transitions to idle
 private volatile ResourceCallback resourceCallback;

 public FragmentIdlingResource(FragmentManager fragmentManager, String tag) {
 mFragmentManager = fragmentManager;
 mTag = tag;
 }

 @Override
 public String getName() {
 return FragmentIdlingResource.class.getName() + ":" + mTag;
 }

 @Override
 public boolean isIdleNow() {
 //simple check, if your fragment is added, then your app has became idle
 boolean idle = (mFragmentManager.findFragmentByTag(mTag) != null);
 if (idle) {
 //IMPORTANT: make sure you call onTransitionToIdle
 resourceCallback.onTransitionToIdle();
 }
 return idle;
 }

 @Override
 public void registerIdleTransitionCallback(ResourceCallback resourceCallback) {
 this.resourceCallback = resourceCallback;
 }
}
```

Now that you have your IdlingResource written, you need to use it somewhere right?

## Usage

Let us skip the entire test class setup and just look how a test case would look like:

```

@Test
public void testSomeFragmentText() {
 mActivityTestRule.launchActivity(null);

 //创建空闲资源
 IdlingResource fragmentLoadedIdlingResource = new
 FragmentIdlingResource(mActivityTestRule.getActivity().getSupportFragmentManager(),
 SomeFragmentText.TAG);
 //注册空闲资源以便Espresso等待它
 Espresso.registerIdlingResources(idlingResource1);
 onView(withId(R.id.txtHelloWorld)).check(matches(withText(helloWorldText)));

 //让我们清理一下
 Espresso.unregisterIdlingResources(fragmentLoadedIdlingResource);
}

```

## 与JUnit规则结合使用

这并不难；你也可以将空闲资源以JUnit测试规则的形式应用。例如，假设你有一个包含Volley的SDK，并且你希望Espresso等待它。与其在每个测试用例中处理或在setup中应用，不如创建一个JUnit规则，然后写：

```

@Rule
public final SDKIdlingRule mSdkIdlingRule = new SDKIdlingRule(SDKInstanceHolder.getInstance());

```

既然这是一个示例，不要当真；这里的所有代码都是虚构的，仅用于演示目的：

```

public class SDKIdlingRule implements TestRule {
 //你编写的空闲资源，用于检查Volley是否空闲
 private VolleyIdlingResource mVolleyIdlingResource;
 //你需要从Volley请求队列中获取以提供给空闲资源
 private RequestQueue mRequestQueue;

 //使用规则时从您的SDK中提取请求队列
 public SDK空闲规则(SDK类 sdkClass) {
 mRequestQueue = 获得Volley请求队列(sdkClass);
 }

 private RequestQueue 获得Volley请求队列(SDK类 sdkClass) {
 return sdkClass.获得Volley请求队列();
 }

 @Override
 public Statement apply(final Statement base, Description description) {
 return new Statement() {
 @Override
 public void 评估() throws Throwable {
 //注册空闲资源
 mVolley空闲资源 = new Volley空闲资源(mRequestQueue);
 Espresso.注册空闲资源(mVolley空闲资源);
 try {
 base.evaluate();
 } finally {
 if (mVolley空闲资源 != null) {
 //测试结束时注销资源
 Espresso.注销空闲资源(mVolley空闲资源);
 }
 }
 }
 };
 }
}

```

```

@Test
public void testSomeFragmentText() {
 mActivityTestRule.launchActivity(null);

 //creating the idling resource
 IdlingResource fragmentLoadedIdlingResource = new
 FragmentIdlingResource(mActivityTestRule.getActivity().getSupportFragmentManager(),
 SomeFragmentText.TAG);
 //registering the idling resource so espresso waits for it
 Espresso.registerIdlingResources(idlingResource1);
 onView(withId(R.id.txtHelloWorld)).check(matches(withText(helloWorldText)));

 //lets cleanup after ourselves
 Espresso.unregisterIdlingResources(fragmentLoadedIdlingResource);
}

```

## Combination with JUnit rule

This is not to hard; you can also apply the idling resource in form of a JUnit test rule. For example, let us say that you have some SDK that contains Volley in it and you want Espresso to wait for it. Instead of going through each test case or applying it in setup, you could create a JUnit rule and just write:

```

@Rule
public final SDKIdlingRule mSdkIdlingRule = new SDKIdlingRule(SDKInstanceHolder.getInstance());

```

Now since this is an example, don't take it for granted; all code here is imaginary and used only for demonstration purposes:

```

public class SDKIdlingRule implements TestRule {
 //idling resource you wrote to check is volley idle or not
 private VolleyIdlingResource mVolleyIdlingResource;
 //request queue that you need from volley to give it to idling resource
 private RequestQueue mRequestQueue;

 //when using the rule extract the request queue from your SDK
 public SDKIdlingRule(SDKClass sdkClass) {
 mRequestQueue = getVolleyRequestQueue(sdkClass);
 }

 private RequestQueue getVolleyRequestQueue(SDKClass sdkClass) {
 return sdkClass.getVolleyRequestQueue();
 }

 @Override
 public Statement apply(final Statement base, Description description) {
 return new Statement() {
 @Override
 public void evaluate() throws Throwable {
 //registering idling resource
 mVolleyIdlingResource = new VolleyIdlingResource(mRequestQueue);
 Espresso.registerIdlingResources(mVolleyIdlingResource);
 try {
 base.evaluate();
 } finally {
 if (mVolleyIdlingResource != null) {
 //deregister the resource when test finishes
 Espresso.unregisterIdlingResources(mVolleyIdlingResource);
 }
 }
 }
 };
 }
}

```

}

}

# 第253章：使用

JUnit进行Android单元测试

## 第253.1节：将业务逻辑移出Android组件

本地JVM单元测试的很多价值来自于你设计应用程序的方式。你必须以一种能够将业务逻辑与Android组件解耦的方式来设计它。这里是一个使用模型-视图-主持人模式的示例。让我们通过实现一个只接受用户名和密码的基本注册界面来练习这个方法。我们的Android应用负责验证用户提供的用户名不为空，且密码至少包含八个字符并且至少包含一个数字。如果用户名/密码有效，我们执行注册API调用，否则显示错误信息。

业务逻辑高度耦合于Android组件的示例。

```
public class LoginActivity extends Activity{
 ...
 private void onSubmitButtonClicked(){
 String username = findViewById(R.id.username).getText().toString();
 String password = findViewById(R.id.password).getText().toString();
 boolean usernameValid = username != null && username.trim().length() != 0;
 boolean passwordValid = password != null && password.trim().length() >= 8 &&
 password.matches(".*\\d+.*");
 if(usernameValid && passwordValid){
 performSignUpApiCall(username, password);
 } else {
 displayInvalidCredentialsErrorMessage();
 }
 }
}
```

业务逻辑与Android组件解耦的示例。

这里我们在一个名为 LoginContract 的类中定义，将包含我们各个类之间的各种交互。

```
public interface LoginContract {
 public interface View {
 executeSignUpApiCall(String username, String password);
 displayInvalidCredentialsErrorMessage();
 }
 public interface Presenter {
 void validateUserCredentials(String username, String password);
 }
}
```

我们的登录活动大致相同，只是我们移除了必须知道如何验证用户注册表单（我们的业务逻辑）的责任。登录活动现在将依赖我们新的登录演示者来执行验证。

```
public class LoginActivity extends Activity implements LoginContract.View{
 private LoginContract.Presenter presenter;

 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 presenter = new LoginPresenter(this);
 ...
 }
```

# Chapter 253: Unit testing in Android with JUnit

## Section 253.1: Moving Business Logic Out of Android Components

A lot of the value from local JVM unit tests comes from the way you design your application. You have to design it in such a way where you can decouple your business logic from your Android Components. Here is an example of such a way using the [Model-View-Presenter pattern](#). Lets practice this out by implementing a basic sign up screen that only takes a username and password. Our Android app is responsible for validating that the username the user supplies is not blank and that the password is at least eight characters long and contains at least one digit. If the username/password is valid we perform our sign up api call, otherwise we display an error message.

Example where business logic is highly coupled with Android Component.

```
public class LoginActivity extends Activity{
 ...
 private void onSubmitButtonClicked(){
 String username = findViewById(R.id.username).getText().toString();
 String password = findViewById(R.id.password).getText().toString();
 boolean usernameValid = username != null && username.trim().length() != 0;
 boolean passwordValid = password != null && password.trim().length() >= 8 &&
 password.matches(".*\\d+.*");
 if(usernameValid && passwordValid){
 performSignUpApiCall(username, password);
 } else {
 displayInvalidCredentialsErrorMessage();
 }
 }
}
```

Example where business logic is decoupled from Android Component.

Here we define in a single class, LoginContract, that will house the various interactions between our various classes.

```
public interface LoginContract {
 public interface View {
 executeSignUpApiCall(String username, String password);
 displayInvalidCredentialsErrorMessage();
 }
 public interface Presenter {
 void validateUserCredentials(String username, String password);
 }
}
```

Our LoginActivity is for the most part the same except that we have removed the responsibility of having to know how to validate a user's sign up form (our business logic). The LoginActivity will now rely on our new LoginPresenter to perform validation.

```
public class LoginActivity extends Activity implements LoginContract.View{
 private LoginContract.Presenter presenter;

 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 presenter = new LoginPresenter(this);
 ...
 }
```

```

}
...

private void 当提交按钮被点击(){
 String 用户名 = findViewById(R.id.用户名).getText().toString();
 String 密码 = findViewById(R.id.密码).getText().toString();
 演示者验证用户凭证(用户名, 密码);
}
...
}

```

现在您的业务逻辑将驻留在新的 LoginPresenter 类中。

```

public class LoginPresenter implements LoginContract.Presenter{
 private LoginContract.View view;

 public LoginPresenter(LoginContract.View view){
 this.view = view;
 }

 public void validateUserCredentials(String username, String password){
 boolean isUsernameValid = username != null && username.trim().length() != 0;
 boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
 if(用户名有效 && 密码有效){
view.performSignUpApiCall(username, password);
 } else {
view.displayInvalidCredentialsErrorMessage();
 }
 }
}

```

现在我们可以针对新的 LoginPresenter 类创建本地 JVM 单元测试。

```

public class LoginPresenterTest {

 @Mock
 LoginContract.View view;

 private LoginPresenter presenter;

 @Before
 public void setUp() throws Exception {
 MockitoAnnotations.initMocks(this);
 presenter = new 登录演示者(view);
 }

 @Test
 public void test_validateUserCredentials_userDidNotEnterUsername_displayErrorMessage() throws
Exception {
 String 用户名 = "";
 String 密码 = "kingslayer1";
presenter.validateUserCredentials(用户名, 密码);
 Mockito.verify(view). 显示无效凭证错误信息();
 }

 @Test
 public void
test_validateUserCredentials_userEnteredFourLettersAndOneDigitPassword_displayErrorMessage() throws
Exception {
 String 用户名 = "杰米·兰尼斯特";
}

```

```

}
...

private void onSubmitButtonClicked(){
 String username = findViewById(R.id.username).getText().toString();
 String password = findViewById(R.id.password).getText().toString();
 presenter.validateUserCredentials(username, password);
}
...
}

```

Now your business logic will reside in your new LoginPresenter class.

```

public class LoginPresenter implements LoginContract.Presenter{
 private LoginContract.View view;

 public LoginPresenter(LoginContract.View view){
 this.view = view;
 }

 public void validateUserCredentials(String username, String password){
 boolean isUsernameValid = username != null && username.trim().length() != 0;
 boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
 if(isUsernameValid && isPasswordValid){
 view.performSignUpApiCall(username, password);
 } else {
 view.displayInvalidCredentialsErrorMessage();
 }
 }
}

```

And now we can create local JVM unit tests against your new LoginPresenter class.

```

public class LoginPresenterTest {

 @Mock
 LoginContract.View view;

 private LoginPresenter presenter;

 @Before
 public void setUp() throws Exception {
 MockitoAnnotations.initMocks(this);
 presenter = new LoginPresenter(view);
 }

 @Test
 public void test_validateUserCredentials_userDidNotEnterUsername_displayErrorMessage() throws
Exception {
 String username = "";
 String password = "kingslayer1";
 presenter.validateUserCredentials(username, password);
 Mockito.verify(view). displayInvalidCredentialsErrorMessage();
 }

 @Test
 public void
test_validateUserCredentials_userEnteredFourLettersAndOneDigitPassword_displayErrorMessage() throws
Exception {
 String username = "Jaime Lannister";
}

```

```

 String 密码 = "king1";
presenter.validateUserCredentials(用户名, 密码);
 Mockito.verify(view). 显示无效凭证错误信息();
}

@Test
public void
test_validateUserCredentials_userEnteredNineLettersButNoDigitsPassword_displayErrorMessage() throws
Exception {
 String username = "Jaime Lannister";
 String password = "kingslayer";
presenter.validateUserCredentials(用户名, 密码);
 Mockito.verify(view). 显示无效凭证错误信息();
}

@Test
public void
 test_validateUserCredentials_userEnteredNineLettersButOneDigitPassword_performApiCallToSignUpUser()
throws Exception {
 String username = "Jaime Lannister";
 String password = "kingslayer1";
presenter.validateUserCredentials(username, password);
 Mockito.verify(view).performSignUpApiCall(username, password);
}
}

```

正如你所见，当我们将业务逻辑从 LoginActivity 中提取并放入 LoginPresenter POJO 后，我们现在可以针对业务逻辑创建本地 JVM 单元测试。

需要注意的是，我们架构变更还有其他多种影响，比如我们更接近于遵循每个类单一职责原则，增加了额外的类等。这些只是我选择通过 MVP 风格进行解耦的副作用。MVP 只是实现这一目标的一种方式，但你也可以考虑其他替代方案，比如 MVVM。你只需选择最适合你的系统。

## 第253.2节：创建本地单元测试

将你的测试类放在这里：`/src/test/<pkg_name>/`

### 示例测试类

```

public class ExampleUnitTest {
 @Test
 public void addition_isCorrect() throws Exception {
 int a=4, b=5, c;
c = a + b;
assertEquals(9, c); // 该测试通过
 assertEquals(10, c); // 测试失败
 }
}

```

### 解析

```

public class ExampleUnitTest {
...
}

```

测试类，你可以创建多个测试类并将它们放在测试包中。

`@Test`

```

 String password = "king1";
presenter.validateUserCredentials(username, password);
 Mockito.verify(view). displayInvalidCredentialsErrorMessage();
}

@Test
public void
test_validateUserCredentials_userEnteredNineLettersButNoDigitsPassword_displayErrorMessage() throws
Exception {
 String username = "Jaime Lannister";
 String password = "kingslayer";
presenter.validateUserCredentials(username, password);
 Mockito.verify(view). displayInvalidCredentialsErrorMessage();
}

@Test
public void
 test_validateUserCredentials_userEnteredNineLettersButOneDigitPassword_performApiCallToSignUpUser()
throws Exception {
 String username = "Jaime Lannister";
 String password = "kingslayer1";
presenter.validateUserCredentials(username, password);
 Mockito.verify(view).performSignUpApiCall(username, password);
}
}

```

As you can see, when we extracted our business logic out of the LoginActivity and placed it in the LoginPresenter POJO. We can now create local JVM unit tests against our business logic.

It should be noted that there are various other implications from our change in architecture such as we are close to adhering to each class having a single responsibility, additional classes, etc. These are just side effects of the way I choose to go about performing this decoupling via the MVP style. MVP is just one way to go about this but there are other alternatives that you may want to look at such as MVVM. You just have to pick the best system that works for you.

## Section 253.2: Creating Local unit tests

Place your test classes here: `/src/test/<pkg_name>/`

### Example test class

```

public class ExampleUnitTest {
 @Test
 public void addition_isCorrect() throws Exception {
 int a=4, b=5, c;
c = a + b;
assertEquals(9, c); // This test passes
 assertEquals(10, c); // Test fails
 }
}

```

### Breakdown

```

public class ExampleUnitTest {
...
}

```

The test class, you can create several test classes and place them inside the test package.

`@Test`

```
public void addition_isCorrect() {
 ...
}
```

测试方法，可以在测试类中创建多个测试方法。

注意注解 @Test。

Test 注解告诉 JUnit，附加该注解的 public void 方法可以作为测试用例运行。

还有其他一些有用的注解，如@Before、@After等。此页面是一个很好的起点。

```
assertEquals(9, c); // 该测试通过
assertEquals(10, c); // 测试失败
```

这些方法是Assert类的成员。其他一些有用的方法有assertFalse()、assertNotNull()、assertTrue等。这里有一个详细的[解释](#)。

#### JUnit 测试的注解信息：

**@Test:** Test 注解告诉 JUnit，附加该注解的 public void 方法可以作为测试用例运行。  
要运行该方法，JUnit 首先构造该类的新实例，然后调用被注解的方法。

**@Before:** 编写测试时，通常会发现多个测试需要在运行前创建相似的对象。用@Before注解一个 public void 方法，会使该方法在测试方法之前运行。

**@After:** 如果你在 Before 方法中分配了外部资源，则需要在测试运行后释放它们。

用@After注解一个 public void 方法，会使该方法在测试方法之后运行。即使 Before 或 Test 方法抛出异常，所有@After 方法也保证会被执行。

#### 在 Android Studio 中快速创建测试类

- 将光标放在要为其创建测试类的类名上。
- 按下 Alt + Enter (Windows)。
- 选择“创建测试”，按回车键。
- 选择要为其创建测试方法的方法，点击确定。
- 选择要创建测试类的目录。
- 完成，这就是你的第一个测试。

#### 提示：在 Android Studio 中轻松执行测试

- 右键点击测试包。
- 选择运行“测试包中的测试...”
- 包中的所有测试将一次性执行。

## 第253.3节：JUnit入门

### 设置

要开始使用JUnit对您的Android项目进行单元测试，您需要将JUnit依赖项添加到项目中，并且需要创建一个测试源集，该源集将包含单元测试的源代码。使用Android Studio创建的项目通常已经包含JUnit依赖项和测试源集

```
public void addition_isCorrect() {
 ...
}
```

The test method, several test methods can be created inside a test class.

Notice the annotation @Test.

The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.

There are several other useful annotations like @Before, @After etc. [This page](#) would be a good place to start.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

These methods are member of the **Assert** class. Some other useful methods are assertFalse(), assertNotNull(), assertTrue etc. Here's an elaborate [Explanation](#).

#### Annotation Information for JUnit Test:

**@Test:** The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.  
To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method.

**@Before:** When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before the Test method.

**@After:** If you allocate external resources in a Before method you need to release them after the test runs.  
Annotating a public void method with @After causes that method to be run after the Test method. All @After methods are guaranteed to run even if a Before or Test method throws an exception

#### Tip Quickly create test classes in Android Studio

- Place the cursor on the class name for which you want to create a test class.
- Press Alt + Enter (Windows).
- Select Create Test, hit Return.
- Select the methods for which you want to create test methods, click OK.
- Select the directory where you want to create the test class.
- You're done, this what you get is your first test.

#### Tip Easily execute tests in Android Studio

- Right click test the package.
- Select Run 'Tests in ...'
- All the tests in the package will be executed at once.

## Section 253.3: Getting started with JUnit

### Setup

To start unit testing your Android project using JUnit you need to add the JUnit dependency to your project and you need to create a test source-set which is going to contain the source code for the unit tests. Projects created with Android Studio often already include the JUnit dependency and the test source-set

在模块的build.gradle文件中的dependencies闭包内添加以下行：

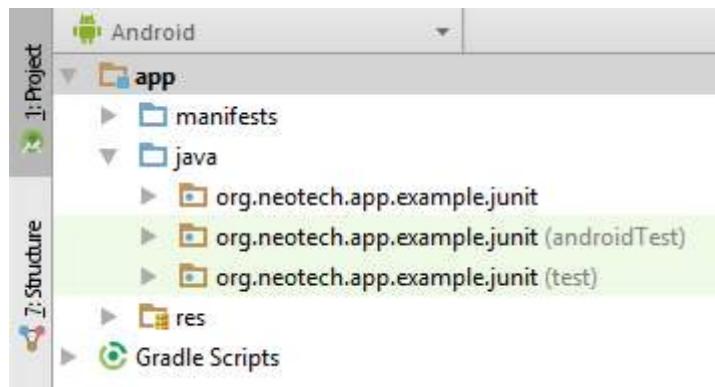
```
testCompile 'junit:junit:4.12'
```

JUnit测试类位于名为test的特殊源集中。如果该源集不存在，您需要自己创建一个新文件夹。默认的Android Studio（基于Gradle）项目的文件夹结构如下：

```
<project-root-folder>
/app (模块根文件夹)
 /build
 /libs
 /src
/main (源代码)
 /test (单元测试源代码)
 /androidTest (仪器测试源代码)
 build.gradle (模块gradle文件)
/build
/gradle
build.gradle (项目gradle文件)
 gradle.properties
 gradlew
 gradlew.bat
local.properties
 settings.gradle (gradle设置)
```

如果你的项目没有/app/src/test文件夹，你需要自己创建它。在 test文件夹内，你还需要一个 java文件夹（如果不存在则创建）。test源代码集中的java文件夹应包含与你的 main源代码集相同的包结构。

如果设置正确，你的项目结构（在Android Studio的Android视图中）应如下所示：



注意：你不一定需要有 androidTest源代码集，这个源代码集通常出现在由Android Studio创建的项目中，此处仅作参考。

## 编写测试

### 1. 在test源代码集中创建一个新类。

在项目视图中右键点击测试源代码集，选择新建 > Java 类。

最常用的命名模式是使用你要测试的类名，并在后面加上Test。所以  
StringUtilities变成StringUtilitiesTest。

### 2. 添加@RunWith注解

需要@RunWith注解以使JUnit运行我们将在测试类中定义的测试。默认的JUnit运行器（针对JUnit 4）是BlockJUnit4ClassRunner，但与其直接使用这个运行器

Add the following line to your module build.gradle file within the dependencies Closure:

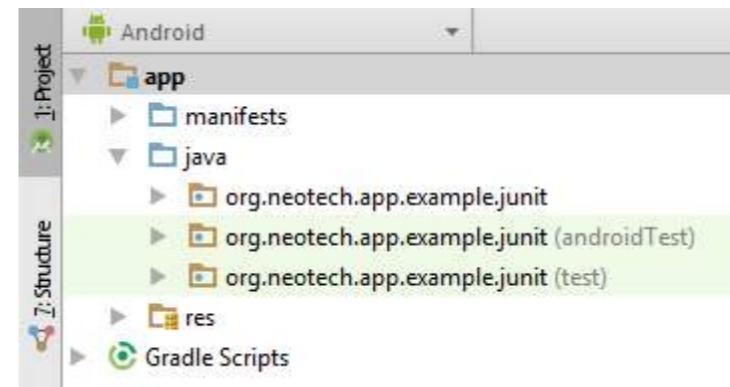
```
testCompile 'junit:junit:4.12'
```

JUnit test classes are located in a special source-set named test. If this source-set does not exist you need to create a new folder yourself. The folder structure of a default Android Studio (Gradle based) project looks like this:

```
<project-root-folder>
/app (module root folder)
 /build
 /libs
 /src
 /main (source code)
 /test (unit test source code)
 /androidTest (instrumentation test source code)
 build.gradle (module gradle file)
/build
/gradle
build.gradle (project gradle file)
 gradle.properties
 gradlew
 gradlew.bat
local.properties
 settings.gradle (gradle settings)
```

If your project doesn't have the /app/src/test folder you need to create it yourself. Within the test folder you also need a java folder (create it if it doesn't exist). The java folder in the test source set should contain the same package structure as your main source-set.

If setup correctly your project structure (in the Android view in Android Studio) should look like this:



Note: You don't necessarily need to have the androidTest source-set, this source-set is often found in projects created by Android Studio and is included here for reference.

## Writing a test

### 1. Create a new class within the test source-set.

Right click the test source-set in the project view choose New > Java class.

The most used naming pattern is to use the name of the class you're going to test with Test added to it. So StringUtilities becomes StringUtilitiesTest.

### 2. Add the @RunWith annotation

The @RunWith annotation is needed in order to make JUnit run the tests we're going to define in our test class. The default JUnit runner (for JUnit 4) is the BlockJUnit4ClassRunner but instead of using this running

更方便的是使用别名JUnit4，它是默认JUnit运行器的简写。

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {
}
```

### 3. 创建一个测试

单元测试本质上就是一个方法，在大多数情况下，运行时不应失败。换句话说，它不应抛出异常。在测试方法内部，你几乎总能找到检查特定条件是否满足的断言。如果断言失败，它会抛出异常，导致方法/测试失败。测试方法总是用@Test注解标注。没有这个注解，JUnit不会自动运行测试。

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

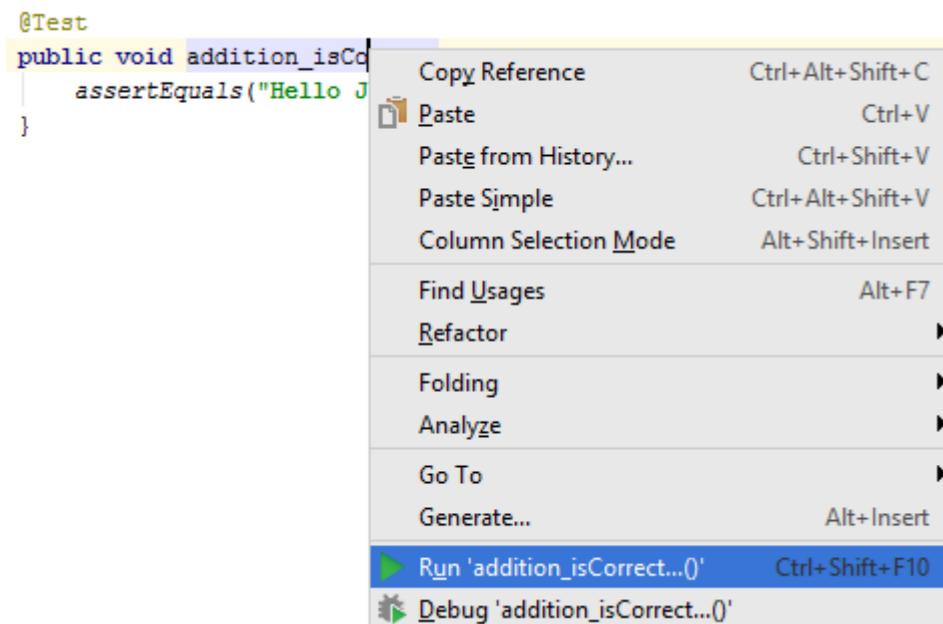
 @Test
 public void addition_isCorrect() throws Exception {
 assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
 }
}
```

注意：与标准的Java方法命名规范不同，单元测试方法名通常包含下划线。

## 运行测试

### 1. 方法

要运行单个测试方法，可以右键点击该方法，然后点击运行'addition\_isCorrect()'，或者使用键盘快捷键ctrl+shift+f10。



如果一切设置正确，JUnit会开始运行该方法，你应该能在Android Studio中看到以下界面：

directly its more convenient to use the alias JUnit4 which is a shorthand for the default JUnit runner.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {
}
```

### 3. Create a test

A unit test is essentially just a method which, in most cases, should not fail if run. In other words it should not throw an exception. Inside a test method you will almost always find assertions that check if specific conditions are met. If an assertion fails it throws an exception which causes the method/test to fail. A test method is always annotated with the @Test annotation. Without this annotation JUnit won't automatically run the test.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

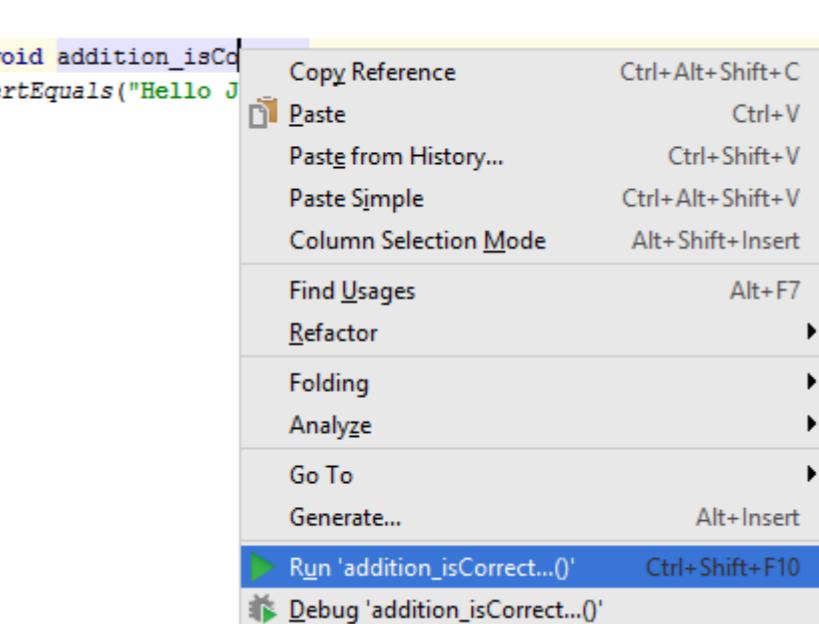
 @Test
 public void addition_isCorrect() throws Exception {
 assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
 }
}
```

*Note: unlike the standard Java method naming convention unit test method names do often contain underscores.*

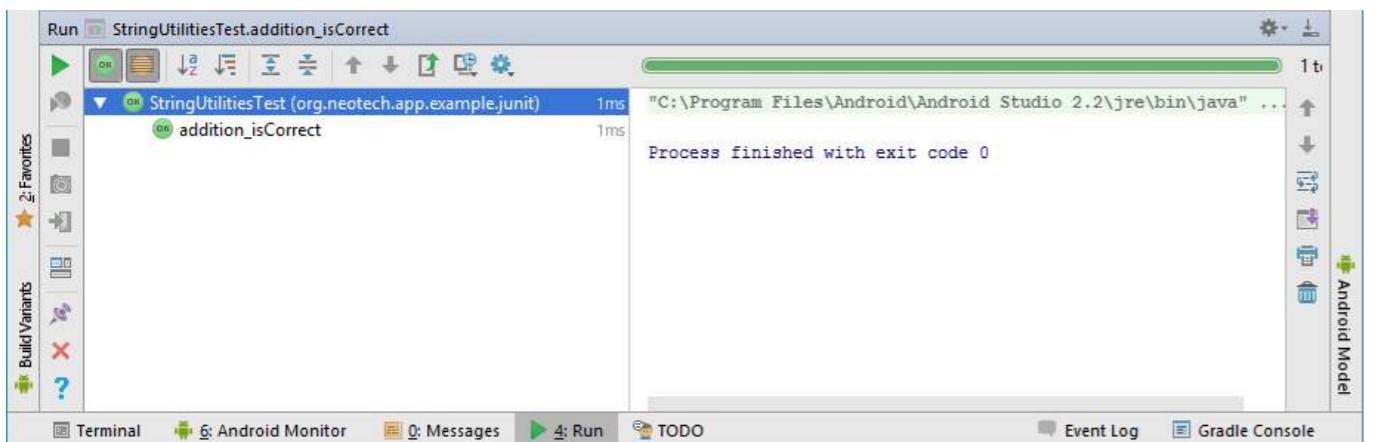
## Running a test

### 1. Method

To run a single test method you can right click the method and click Run 'addition\_isCorrect()' or use the keyboard shortcut ctrl+shift+f10.



If everything is setup correctly JUnit starts running the method and you should see the following interface within Android Studio:



## 2.类

你也可以通过在项目视图中右键点击某个类，然后点击运行'StringUtilitiesTest'来运行该类中定义的所有测试，或者如果你已经在项目视图中选中了该类，可以使用快捷键ctrl+shift+f10。

## 3.包（全部）

如果你想运行项目或某个包中定义的所有测试，只需右键点击该包，然后点击运行...，就像运行单个类中定义的所有测试一样。

## 第253.4节：异常

JUnit也可以用来测试某个方法是否会针对给定输入抛出特定异常。

在这个例子中，我们将测试以下方法是否在布尔格式（输入）无法识别/未知时确实抛出异常：

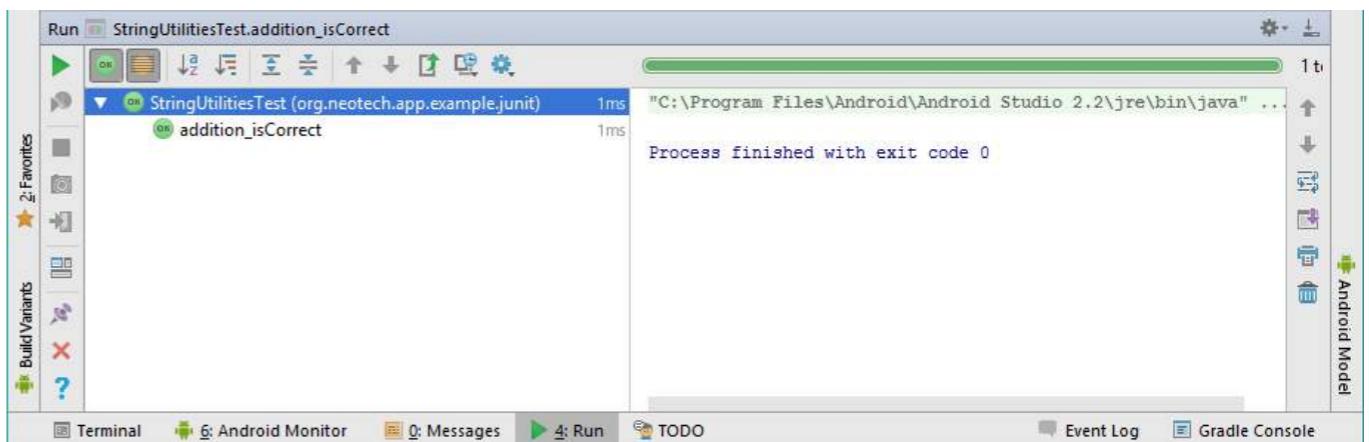
```
public static boolean parseBoolean(@NotNull String raw) throws IllegalArgumentException{
 raw = raw.toLowerCase().trim();
 switch (raw) {
 case "t": case "yes": case "1": case "true":
 return true;
 case "f": case "no": case "0": case "false":
 return false;
 default:
 throw new IllegalArgumentException("未知的布尔格式: " + raw);
 }
}
```

通过向 @Test 注解添加 expected 参数，可以定义预期抛出的异常。如果该异常未发生，单元测试将失败；如果异常确实抛出，则测试成功：

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
 StringUtilities.parseBoolean("Hello JUnit");
}
```

这种方法效果很好，但它限制了方法内只能有一个测试用例。有时你可能想在一个方法中测试多个用例。常用的解决方法是使用 try-catch 块和 Assert.fail() 方法：

```
@Test
public void parseBoolean_parsesInvalidFormats_throwsException(){
```



## 2. Class

You can also run all the tests defined in a single class, by right clicking the class in the project view and clicking Run 'StringUtilitiesTest' or use the keyboard shortcut ctrl+shift+f10 if you have selected the class in the project view.

## 3. Package (everything)

If you want to run all the tests defined in the project or in a package you can just right click the package and click Run ... just like you would run all the tests defined in a single class.

## Section 253.4: Exceptions

JUnit can also be used to test if a method throws a specific exception for a given input.

In this example we will test if the following method really throws an exception if the Boolean format (input) is not recognized/unknown:

```
public static boolean parseBoolean(@NotNull String raw) throws IllegalArgumentException{
 raw = raw.toLowerCase().trim();
 switch (raw) {
 case "t": case "yes": case "1": case "true":
 return true;
 case "f": case "no": case "0": case "false":
 return false;
 default:
 throw new IllegalArgumentException("Unknown boolean format: " + raw);
 }
}
```

By adding the expected parameter to the @Test annotation, one can define which exception is expected to be thrown. The unit test will fail if this exception does not occur, and succeed if the exception is indeed thrown:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
 StringUtilities.parseBoolean("Hello JUnit");
}
```

This works well, however, it does limit you to just a single test case within the method. Sometimes you might want to test multiple cases within a single method. A technique often used to overcome this limitation is using try-catch blocks and the `Assert.fail()` method:

```
@Test
public void parseBoolean_parsesInvalidFormats_throwsException(){
```

```

try {
StringUtilities.parseBoolean("Hello!");
 fail("预期抛出 IllegalArgumentException");
} catch(IllegalArgumentException e){
}

try {
StringUtilities.parseBoolean("JUnit!");
 fail("预期抛出 IllegalArgumentException");
} catch(IllegalArgumentException e){
}

```

注意：有些人认为在单元测试中测试多个用例是不好的做法。

## 第253.5节：静态导入

JUnit 定义了相当多的 assertEquals 方法，至少每种基本类型都有一个，且有一个用于对象。这些方法默认情况下不能直接调用，应该这样调用：  
**Assert.assertEquals**。但由于这些方法使用频繁，人们几乎总是使用静态导入，这样方法就可以像是类本身的一部分一样直接使用。

要为 assertEquals 方法添加静态导入，请使用以下导入语句：

```
import static org.junit.Assert.assertEquals;
```

你也可以静态导入所有断言方法，包括 assertArrayEquals、assertNotNull 和 assertFalse 等，使用以下静态导入：

```
import static org.junit.Assert.*;
```

**未使用静态导入：**

```

@Test
public void addition_isCorrect(){
 Assert.assertEquals(4 , 2 + 2);
}

```

**使用静态导入：**

```

@Test
public void addition_isCorrect(){
 assertEquals(4 , 2 + 2);
}

```

```

try {
 StringUtilities.parseBoolean("Hello!");
 fail("Expected IllegalArgumentException");
} catch(IllegalArgumentException e){
}

try {
 StringUtilities.parseBoolean("JUnit!");
 fail("Expected IllegalArgumentException");
} catch(IllegalArgumentException e){
}

```

*Note: Some people consider it to be bad practice to test more than a single case inside a unit test.*

## Section 253.5: Static import

JUnit defines quite some assertEquals methods at least one for each primitive type and one for Objects is available. These methods are by default not directly available to call and should be called like this:  
**Assert.assertEquals**. But because these methods are used so often people almost always use a static import so that the method can be directly used as if it is part of the class itself.

To add a static import for the assertEquals method use the following import statement:

```
import static org.junit.Assert.assertEquals;
```

You can also static import all assert methods including the assertArrayEquals, assertNotNull and assertFalse etc. using the following static import:

```
import static org.junit.Assert.*;
```

**Without static import:**

```

@Test
public void addition_isCorrect(){
 Assert.assertEquals(4 , 2 + 2);
}

```

**With static import:**

```

@Test
public void addition_isCorrect(){
 assertEquals(4 , 2 + 2);
}

```

## 第254章：使用

UIAutomator进行应用间UI测试

### 第254.1节：准备项目并编写第一个 UIAutomator测试

将所需库添加到Android模块的build.gradle的依赖部分：

```
android {
 ...
 defaultConfig {
 ...
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
 }
}

dependencies {
 ...
 androidTestCompile 'com.android.support.test:runner:0.5'
 androidTestCompile 'com.android.support.test:rules:0.5'
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
 androidTestCompile 'com.android.support:support-annotations:23.4.0'
}
```

□注意，版本号当然可能会有所不同。

在此同步更改后。

然后在 androidTest 文件夹内添加一个新的 Java 类：

```
public class InterAppTest extends InstrumentationTestCase {

 private UiDevice device;

 @Override
 public void setUp() throws Exception {
 device = UiDevice.getInstance(getInstrumentation());
 }

 public void testPressHome() throws Exception {
 device.pressHome();
 }
}
```

通过右键点击类标签并选择“运行”InterAppTest”，即可执行此测试。

### 第254.2节：使用

UIAutomatorViewer编写更复杂的测试

为了能够编写更复杂的UI测试，需要使用UIAutomatorViewer。该工具位于/tools/，能够截取当前显示视图布局的全屏截图。请参见下图以了解显示内容的示意：

## Chapter 254: Inter-app UI testing with UIAutomator

### Section 254.1: Prepare your project and write the first UIAutomator test

Add the required libraries into the dependencies section of your Android module's build.gradle:

```
android {
 ...
 defaultConfig {
 ...
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
 }
}

dependencies {
 ...
 androidTestCompile 'com.android.support.test:runner:0.5'
 androidTestCompile 'com.android.support.test:rules:0.5'
 androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
 androidTestCompile 'com.android.support:support-annotations:23.4.0'
}
```

□ Note that of course the versions may differ in the mean time.

After this sync with the changes.

Then add a new Java class inside the androidTest folder:

```
public class InterAppTest extends InstrumentationTestCase {

 private UiDevice device;

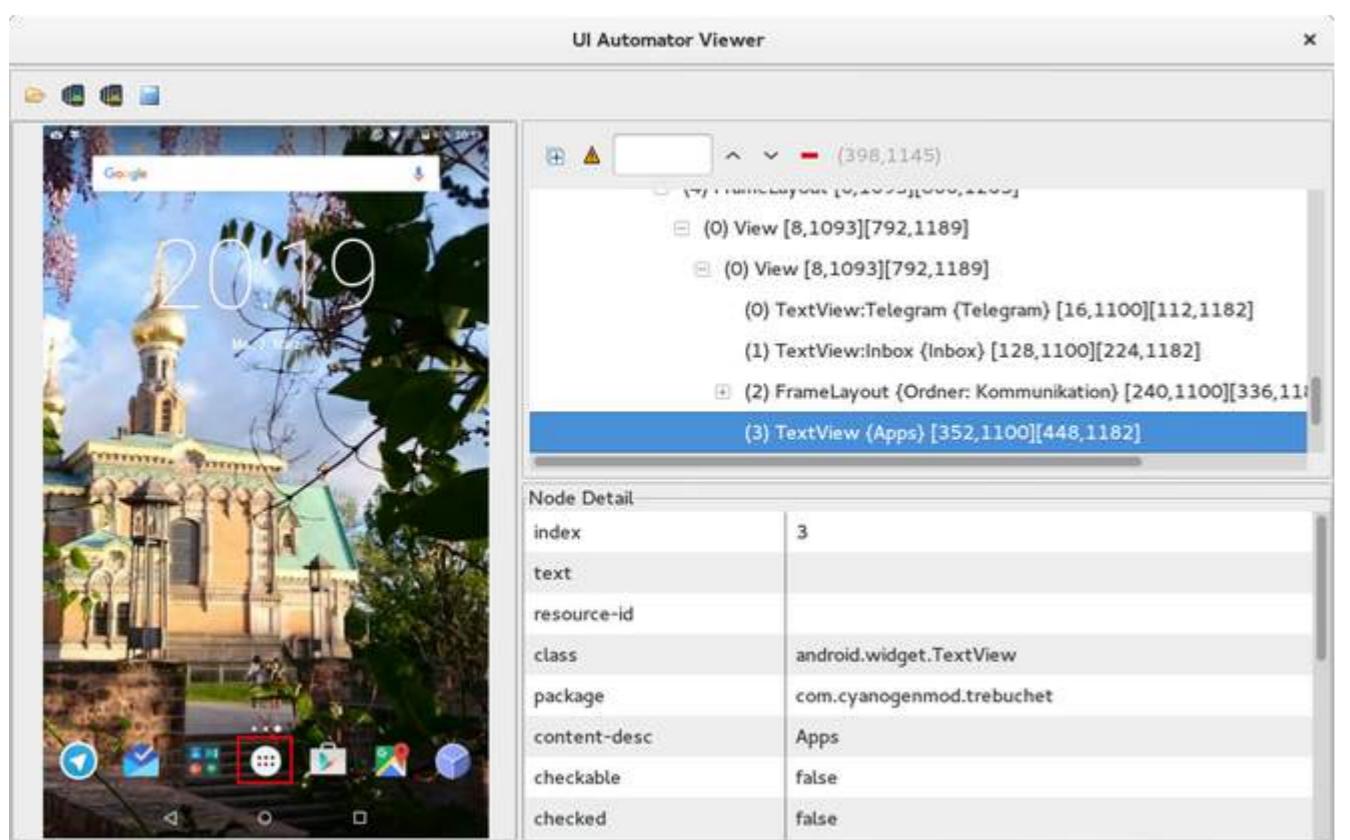
 @Override
 public void setUp() throws Exception {
 device = UiDevice.getInstance(getInstrumentation());
 }

 public void testPressHome() throws Exception {
 device.pressHome();
 }
}
```

By making a right click on the class tab and on "Run "InterAppTest" executes this test.

### Section 254.2: Writing more complex tests using the UIAutomatorViewer

In order to enable writing more complex UI tests the *UIAutomatorViewer* is needed. The tool located at /tools/ makes a fullscreen screenshot including the layouts of the currently displayed views. See the subsequent picture to get an idea of what is shown:



对于UI测试，我们正在寻找resource-id、content-desc或其他用于识别视图并在测试中使用的标识。

通过终端执行uiautomatorviewer。

例如，如果我们现在想点击应用按钮，然后打开某个应用并滑动，这个测试方法可以是这样的：

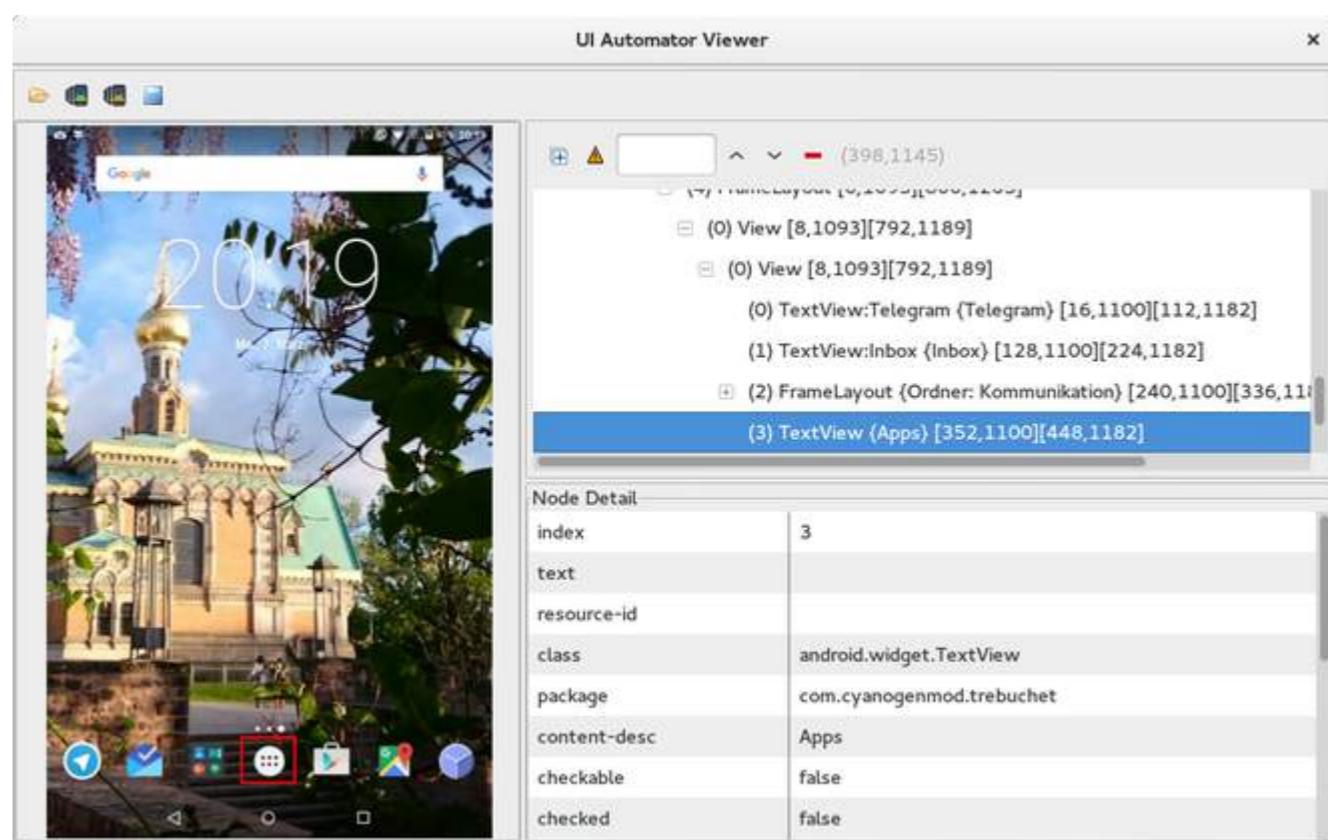
```
public void testOpenMyApp() throws Exception {
 // 唤醒你的设备
 device.wakeUp();

 // 切换到启动器 (如果有应用打开则隐藏)
 device.pressHome();

 // 进入应用菜单 (超时=200毫秒)
 device.wait(Until.hasObject(By.desc("Apps")), 200);
 UiObject2 appsButton = device.findObject(By.desc("Apps"));
 assertNotNull(appsButton);
 appsButton.click();

 // 进入某个应用 (超时=200毫秒)
 device.wait(Until.hasObject(By.desc("MyApplication")), 200);
 UiObject2 someAppIcon = device.findObject(By.desc("MyApplication"));
 assertNotNull(someAppIcon);
 someAppIcon.click();

 // 执行滑动操作 (steps=20 表示0.1秒)
 device.swipe(200, 1200, 1300, 1200, 20);
 assertTrue(isSomeConditionTrue)
}
```



For the UI tests we are looking for *resource-id*, *content-desc* or something else to identify a view and use it inside our tests.

The *uiautomatorviewer* is executed via terminal.

If we now for instance want to click on the applications button and then open some app and swipe around, this is how the test method can look like:

```
public void testOpenMyApp() throws Exception {
 // wake up your device
 device.wakeUp();

 // switch to launcher (hide the previous application, if some is opened)
 device.pressHome();

 // enter applications menu (timeout=200ms)
 device.wait(Until.hasObject(By.desc("Apps")), 200);
 UiObject2 appsButton = device.findObject(By.desc("Apps"));
 assertNotNull(appsButton);
 appsButton.click();

 // enter some application (timeout=200ms)
 device.wait(Until.hasObject(By.desc("MyApplication")), 200);
 UiObject2 someAppIcon = device.findObject(By.desc("MyApplication"));
 assertNotNull(someAppIcon);
 someAppIcon.click();

 // do a swipe (steps=20 is 0.1 sec.)
 device.swipe(200, 1200, 1300, 1200, 20);
 assertTrue(isSomeConditionTrue)
}
```

## 第254.3节：创建UIAutomator测试套件

将UIAutomator测试组合成测试套件是件快速的事：

```
package de.androidtest.myapplication;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({InterAppTest1.class, InterAppTest2.class})
public class AppTestSuite {}
```

通过右键点击并运行套件，执行方式类似于单个测试。

## Section 254.3: Creating a test suite of UIAutomator tests

Putting UIAutomator tests together to a test suite is a quick thing:

```
package de.androidtest.myapplication;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({InterAppTest1.class, InterAppTest2.class})
public class AppTestSuite {}
```

Execute similar to a single test by clicking right and run the suite.

# 第255章：Lint警告

## 第255.1节：在xml文件中使用tools:ignore

属性tools:ignore可以在xml文件中用来忽略lint警告。

但用这种方法忽略lint警告大多数情况下是错误的做法。

必须理解并修复lint警告.....只有在你完全理解其含义并有充分理由忽略它时，才可以忽略。

以下是一个合法忽略lint警告的使用场景：

- 你正在开发一个系统应用（使用设备制造商密钥签名）
- 你的应用需要更改设备日期（或执行其他受保护操作）

那么你可以在你的清单文件中这样做：（即请求受保护权限并忽略lint警告

因为你知道在你的情况下该权限会被授予）

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
...
<uses-permission android:name="android.permission.SET_TIME"
 tools:ignore="ProtectedPermissions"/>
```

## 第255.2节：使用gradle配置LintOptions

您可以通过在build.gradle文件中添加lintOptions部分来配置lint：

```
android {
 //.....
 lintOptions {
 // 关闭对指定问题ID的检查
 disable 'TypographyFractions','TypographyQuotes'

 // 开启指定问题ID的检查
 enable 'RtlHardcoded','RtlCompat', 'RtlEnabled'

 // 仅检查指定的问题ID
 check 'NewApi', 'InlinedApi'

 // 设置为 true 以关闭 lint 的分析进度报告
 quiet true

 // 如果为 true, 发现错误时停止 gradle 构建
 abortOnError false

 // 如果为 true, 仅报告错误
 ignoreWarnings true
 }
}
```

你可以针对特定变体运行 lint（见下文），例如 ./gradlew lintRelease，或者针对所有变体（./gradlew lint），此时会生成一份报告，描述某个问题适用于哪些具体变体。

# Chapter 255: Lint Warnings

## Section 255.1: Using tools:ignore in xml files

The attribute tools:ignore can be used in xml files to dismiss lint warnings.

**BUT dismissing lint warnings with this technique is most of the time the wrong way to proceed.**

A lint warning must be understood and fixed... it can be ignored if and only if you have a full understanding of its meaning and a strong reason to ignore it.

Here is a use case where it legitimate to ignore a lint warning:

- You are developing a system-app (signed with the device manufacturer key)
- Your app need to change the device date (or any other protected action)

Then you can do this in your manifest : (i.e. requesting the protected permission and ignoring the lint warning because you know that in your case the permission will be granted)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
...
<uses-permission android:name="android.permission.SET_TIME"
 tools:ignore="ProtectedPermissions"/>
```

## Section 255.2: Configure LintOptions with gradle

You can configure lint by adding a lintOptions section in the build.gradle file:

```
android {
 //.....
 lintOptions {
 // turn off checking the given issue id's
 disable 'TypographyFractions', 'TypographyQuotes'

 // turn on the given issue id's
 enable 'RtlHardcoded', 'RtlCompat', 'RtlEnabled'

 // check *only* the given issue id's
 check 'NewApi', 'InlinedApi'

 // set to true to turn off analysis progress reporting by lint
 quiet true

 // if true, stop the gradle build if errors are found
 abortOnError false

 // if true, only report errors
 ignoreWarnings true
 }
}
```

You can run lint for a specific variant (see below), e.g. ./gradlew lintRelease, or for all variants (./gradlew lint), in which case it produces a report which describes which specific variants a given issue applies to.

请在此处查看所有可用选项的 [DSL 参考](#)。

## 第 255.3 节：配置 Java 和 XML

### 源文件中的 Lint 检查

你可以禁用 Java 和 XML 源文件中的 Lint 检查。

#### 在Java中配置Lint检查

要在Android项目中针对特定的**Java类**或方法禁用Lint检查，请在该Java代码中添加@SuppressLint注解。

示例：

```
@SuppressLint("NewApi")
@Override
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
```

要禁用所有Lint问题的检查：

```
@SuppressLint("all")
```

#### 在XML中配置Lint检查

您可以使用tools:ignore属性来禁用对特定XML文件部分的Lint检查。

例如：

```
tools:ignore="NewApi,StringFormatInvalid"
```

要在XML元素中抑制所有Lint问题的检查，请使用

```
工具:忽略="all"
```

## 第255.4节：如何配置lint.xml文件

您可以在lint.xml文件中指定Lint检查的偏好设置。如果您手动创建此文件，请将其放置在Android项目的根目录中。如果您在Android Studio中配置Lint偏好设置，lint.xml文件会自动创建并添加到您的Android项目中。

示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
 <!-- 配置的问题列表 -->
</lint>
```

通过设置标签中的severity属性值，您可以禁用某个问题的Lint检查或更改该问题的严重级别。

下面的示例显示了lint.xml文件的内容。

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
```

Check here for the [DSL reference for all available options](#).

## Section 255.3: Configuring lint checking in Java and XML source files

You can disable Lint checking from your Java and XML source files.

#### Configuring lint checking in Java

To disable Lint checking specifically for a **Java class** or method in your Android project, add the @SuppressLint annotation to that Java code.

Example:

```
@SuppressLint("NewApi")
@Override
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
```

To disable checking for all Lint issues:

```
@SuppressLint("all")
```

#### Configuring lint checking in XML

You can use the tools:ignore attribute to disable Lint checking for specific sections of your **XML files**.

For example:

```
tools:ignore="NewApi,StringFormatInvalid"
```

To suppress checking for all Lint issues in the XML element, use

```
tools:ignore="all"
```

## Section 255.4: How to configure the lint.xml file

You can specify your Lint checking preferences in the **lint.xml** file. If you are creating this file manually, place it in the root directory of your Android project. If you are configuring Lint preferences in Android Studio, the **lint.xml** file is automatically created and added to your Android project for you.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
 <!-- list of issues to configure -->
</lint>
```

By setting the severity attribute value in the tag, you can disable Lint checking for an issue or change the severity level for an issue.

The following example shows the contents of a **lint.xml** file.

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
```

```

<!-- 在此项目中禁用指定的检查 -->
<issue id="IconMissingDensityFolder" severity="ignore" />

<!-- 在指定文件中忽略ObsoleteLayoutParams问题 -->
<issue id="ObsoleteLayoutParams">
 <ignore path="res/layout/activation.xml" />
 <ignore path="res/layout-xlarge/activation.xml" />

<!-- 忽略指定文件中的UselessLeaf问题 -->
<issue id="UselessLeaf">
 <ignore path="res/layout/main.xml" />

<!-- 将硬编码字符串的严重性改为“错误” -->
<issue id="HardcodedText" severity="error" />
</lint>

```

## 第255.5节：标记抑制警告

在代码中标记某些警告是良好的实践。例如，某些已弃用的方法是你测试或旧版本支持所必需的。但Lint检查会将这些代码标记为警告。为避免此问题，你需要使用注解@SuppressWarnings。

例如，向已弃用方法添加忽略警告。你还需要在注解中写明警告描述：

```

@SuppressLint("deprecation")
public void setAnotherColor (int newColor) {
 getApplicationContext().getResources().getColor(newColor)
}

```

使用此注解你可以忽略所有警告，包括Lint、Android及其他警告。使用SuppressWarnings，有助于正确理解代码！

## 第255.6节：导入资源时无“已弃用”错误

使用 Android API 23 或更高版本时，经常会遇到以下情况：

```

context.getResources().getColor(R.color.colorPrimaryDark);
init();
'getColor(int)' is deprecated more... (Ctrl+F1)

```

这种情况是由于 Android API 在获取资源方面的结构性变化引起的。现在应使用以下函数：

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

但 android.support.v4 库有另一种解决方案。

在 build.gradle 文件中添加以下依赖：

```
com.android.support:support-v4:24.0.0
```

然后支持库中的所有方法都可用：

```

<!-- Disable the given check in this project -->
<issue id="IconMissingDensityFolder" severity="ignore" />

<!-- Ignore the ObsoleteLayoutParams issue in the specified files -->
<issue id="ObsoleteLayoutParams">
 <ignore path="res/layout/activation.xml" />
 <ignore path="res/layout-xlarge/activation.xml" />
</issue>

<!-- Ignore the UselessLeaf issue in the specified file -->
<issue id="UselessLeaf">
 <ignore path="res/layout/main.xml" />
</issue>

<!-- Change the severity of hardcoded strings to "error" -->
<issue id="HardcodedText" severity="error" />
</lint>

```

## Section 255.5: Mark Suppress Warnings

It's good practice to mark some warnings in your code. For example, some deprecated methods is need for your testing, or old support version. But Lint checking will mark that code with warnings. For avoiding this problem, you need use annotation @SuppressWarnings.

For example, add ignoring to warnings to deprecated methods. You need to put warnings description in annotation also:

```

@SuppressLint("deprecation")
public void setAnotherColor (int newColor) {
 getApplicationContext().getResources().getColor(newColor)
}

```

Using this annotation you can ignore all warnings, including Lint, Android, and other. Using Suppress Warnings, helps to understand code correctly!

## Section 255.6: Importing resources without "Deprecated" error

Using the Android API 23 or higher, very often such situation can be seen:

```

context.getResources().getColor(R.color.colorPrimaryDark);
init();
'getColor(int)' is deprecated more... (Ctrl+F1)

```

This situation is caused by the structural change of the Android API regarding getting the resources. Now the function:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

should be used. But the android.support.v4 library has another solution.

Add the following dependency to the build.gradle file:

```
com.android.support:support-v4:24.0.0
```

Then all methods from support library are available:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark);
```

此外，还可以使用支持库中的更多方法：

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
```

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark));
```

Moreover more methods from support library can be used:

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
...
```

# 第256章：性能优化

应用的性能是用户体验的关键因素。尽量避免在UI线程上执行耗时操作等性能不佳的模式，学习如何编写快速且响应灵敏的应用程序。

## 第256.1节：使用ViewHolder模式节省视图查找

尤其是在ListView中，滚动时调用过多的findViewById()可能导致性能问题。通过使用ViewHolder模式，可以节省这些查找操作，提升ListView的性能。

如果列表项只包含一个TextView，创建一个ViewHolder类来存储该实例：

```
static class ViewHolder {
 TextView myTextView;
}
```

在创建列表项时，附加一个ViewHolder对象到列表项：

```
public View getView(int position, View convertView, ViewGroup parent) {
 Item i = getItem(position);
 if(convertView == null) {
 convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_item, parent, false);

 // 创建一个新的 ViewHolder 并保存 TextView 实例
 ViewHolder holder = new ViewHolder();
 holder.myTextView = (TextView)convertView.findViewById(R.id.my_text_view);
 convertView.setTag(holder);
 }

 // 获取 ViewHolder 并使用 TextView
 ViewHolder holder = (ViewHolder)convertView.getTag();
 holder.myTextView.setText(i.getText());

 return convertView;
}
```

使用这种模式，findViewById()只会在创建新的View时调用，ListView可以更高效地回收你的视图。

# Chapter 256: Performance Optimization

Your Apps performance is a crucial element of the user experience. Try to avoid bad performing patterns like doing work on the UI thread and learn how to write fast and responsive apps.

## Section 256.1: Save View lookups with the ViewHolder pattern

Especially in a [ListView](#), you can run into performance problems by doing too many `findViewById()` calls during scrolling. By using the ViewHolder pattern, you can save these lookups and improve your [ListView](#) performance.

If your list item contains a single TextView, create a ViewHolder class to store the instance:

```
static class ViewHolder {
 TextView myTextView;
}
```

While creating your list item, attach a ViewHolder object to the list item:

```
public View getView(int position, View convertView, ViewGroup parent) {
 Item i = getItem(position);
 if(convertView == null) {
 convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_item, parent, false);

 // Create a new ViewHolder and save the TextView instance
 ViewHolder holder = new ViewHolder();
 holder.myTextView = (TextView)convertView.findViewById(R.id.my_text_view);
 convertView.setTag(holder);
 }

 // Retrieve the ViewHolder and use the TextView
 ViewHolder holder = (ViewHolder)convertView.getTag();
 holder.myTextView.setText(i.getText());

 return convertView;
}
```

Using this pattern, `findViewById()` will only be called when a new `View` is being created and the [ListView](#) can recycle your views much more efficiently.

# 第257章：Android内核优化

## 第257.1节：低内存配置

Android现在支持512MB内存的设备。本文件旨在帮助OEM优化和配置Android 4.4以适应低内存设备。其中一些优化足够通用，也可以应用于之前的版本。

### 启用低内存设备标志

我们引入了一个新的API，称为ActivityManager.isLowRamDevice()，供应用程序判断是否应关闭在低内存设备上表现不佳的特定高内存消耗功能。

对于512MB设备，该API预期返回：“true”。可以通过设备makefile中的以下系统属性启用该功能。

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

### 禁用JIT

系统范围内的JIT内存使用量取决于运行的应用程序数量及其代码占用。JIT设定了最大已翻译代码缓存大小，并根据需要访问其中的页面。JIT在典型运行系统中占用大约3M到6M的内存。

大型应用程序往往很快就会用满代码缓存（默认大小为1M）。平均而言，JIT缓存使用量每个应用约为100K到200K字节。减少缓存最大大小可以在一定程度上降低内存使用，但设置过低会导致JIT进入频繁换页的状态。对于内存极低的设备，我们建议完全禁用JIT。

这可以通过在产品的 makefile 中添加以下行来实现：

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

## 第257.2节：如何添加CPU调节器

CPU调节器本身只是一个C文件，位于kernel\_source/drivers/cpufreq/目录下，例如：

cpufreq\_smartass2.c。你需要自己找到调节器（查看你设备的现有内核仓库）但为了成功调用并将此文件编译进内核，你需要做以下更改：

1. 复制你的调节器文件（cpufreq\_govname.c），然后浏览到kernel\_source/drivers/cpufreq目录，粘贴进去。
2. 打开Kconfig（这是配置菜单布局的接口），当添加内核时，你希望它能显示在配置中。你可以通过添加调节器选项来实现。

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq 调节器"
depends on CPU_FREQ
help
governor' - 一个自定义调节器！
```

例如，针对smartassV2。

```
config CPU_FREQ_GOV_SMARTASS2
tristate "'smartassV2' cpufreq 调速器"
```

# Chapter 257: Android Kernel Optimization

## Section 257.1: Low RAM Configuration

Android now supports devices with 512MB of RAM. This documentation is intended to help OEMs optimize and configure Android 4.4 for low-memory devices. Several of these optimizations are generic enough that they can be applied to previous releases as well.

### Enable Low Ram Device flag

We are introducing a new API called ActivityManager.isLowRamDevice() for applications to determine if they should turn off specific memory-intensive features that work poorly on low-memory devices.

For 512MB devices, this API is expected to return: "true" It can be enabled by the following system property in the device makefile.

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

### Disable JIT

System-wide JIT memory usage is dependent on the number of applications running and the code footprint of those applications. The JIT establishes a maximum translated code cache size and touches the pages within it as needed. JIT costs somewhere between 3M and 6M across a typical running system.

The large apps tend to max out the code cache fairly quickly (which by default has been 1M). On average, JIT cache usage runs somewhere between 100K and 200K bytes per app. Reducing the max size of the cache can help somewhat with memory usage, but if set too low will send the JIT into a thrashing mode. For the really low-memory devices, we recommend the JIT be disabled entirely.

This can be achieved by adding the following line to the product makefile:

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

## Section 257.2: How to add a CPU Governor

The CPU governor itself is just 1 C file, which is located in kernel\_source/drivers/cpufreq/, for example: cpufreq\_smartass2.c. You are responsible yourself for find the governor (look in an existing kernel repo for your device) But in order to successfully call and compile this file into your kernel you will have to make the following changes:

1. Copy your governor file (cpufreq\_govname.c) and browse to kernel\_source/drivers/cpufreq, now paste it.
2. and open Kconfig (this is the interface of the config menu layout) when adding a kernel, you want it to show up in your config. You can do that by adding the choice of governor.

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

for example, for smartassV2.

```
config CPU_FREQ_GOV_SMARTASS2
tristate "'smartassV2' cpufreq governor"
```

```
depends on CPU_FREQ
help
'smartassV2' - 一个 "智能" 优化的调速器!
```

除了添加该选项外，你还必须声明该调速器被选为默认调速器的可能性。

```
config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
使用 CPUFreq 调速器 'govname' 作为 默认。
```

例如，针对smartassV2。

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
将 CPUFreq 调速器'smartassV2'用作默认。
```

-找不到合适的位置放置它？只需搜索“CPU\_FREQ\_GOV\_CONSERVATIVE”，然后将代码放在其下方，同样适用于“CPU\_FREQ\_DEFAULT\_GOV\_CONSERVATIVE”

现在 Kconfig 已完成，您可以保存并关闭文件。

3. 仍然在/drivers/cpufreq文件夹中，打开Makefile。在Makefile中，添加与你的相应的CPU 调节器。例如：

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2) += cpufreq_smartass2.o
```

注意不要调用原生的 C 文件，而是调用 O 文件！即编译后的 C 文件。保存文件。

4. 移动到：kernel\_source/includes/linux。现在打开 cpufreq.h 向下滚动直到看到类似内容：

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
extern struct cpufreq_governor cpufreq_gov_ondemand;
#define CPUFREQ_DEFAULT_GOVERNOR (&cpufreq_gov_ondemand)
```

(其他 CPU 调节器也列在那里)

现在添加您选择的 CPU 调节器条目，例如：

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
extern struct cpufreq_governor cpufreq_gov_smartass2;
#define CPUFREQ_DEFAULT_GOVERNOR (&cpufreq_gov_smartass2)
```

保存文件并关闭。

初始的CPU调速器设置现在已完成。当你成功完成所有步骤后，应该能够从菜单 (menuconfig、xconfig、gconfig、nconfig) 中选择你的调速器。菜单中选中后，它将被包含进内核。

提交内容与上述指令几乎相同：添加 smartassV2 和 lulzactive 调速器的提交

```
depends on CPU_FREQ
help
'smartassV2' - a "smart" optimized governor!
```

next to adding the choice, you also must declare the possibility that the governor gets chosen as default governor.

```
config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
 bool "gov_name_lowercase"
 select CPU_FREQ_GOV_GOVNAMEHERE
 help
 Use the CPUFreq governor 'govname' as default.
```

for example, for smartassV2.

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

- can't find the right place to put it? Just search for "CPU\_FREQ\_GOV\_CONSERVATIVE", and place the code beneath, same thing counts for "CPU\_FREQ\_DEFAULT\_GOV\_CONSERVATIVE"

Now that Kconfig is finished you can save and close the file.

3. While still in the /drivers/cpufreq folder, open Makefile. In Makefile, add the line corresponding to your CPU Governor. for example:

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2) += cpufreq_smartass2.o
```

Be ware that you do not call the native C file, but the O file! which is the compiled C file. Save the file.

4. Move to: kernel\_source/includes/linux. Now open cpufreq.h Scroll down until you see something like:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
extern struct cpufreq_governor cpufreq_gov_ondemand;
#define CPUFREQ_DEFAULT_GOVERNOR (&cpufreq_gov_ondemand)
```

(other cpu governors are also listed there)

Now add your entry with the selected CPU Governor, example:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
extern struct cpufreq_governor cpufreq_gov_smartass2;
#define CPUFREQ_DEFAULT_GOVERNOR (&cpufreq_gov_smartass2)
```

Save the file and close it.

The initial CPU Governor setup is now complete. when you've done all steps successfully, you should be able to choose your governor from the menu (menuconfig, xconfig, gconfig, nconfig). Once checked in the menu it will be included to the kernel.

Commit that is nearly the same as above instructions: [Add smartassV2 and lulzactive governor commit](#)

## 第257.3节：I/O 调度器

如果需要，你可以通过添加新的I/O调度器来增强内核。总体来说，调速器和调度器是相同的；它们都提供系统应如何工作的方式。然而，对于调度器来说，重点是输入/输出数据流，除了CPU设置。I/O调度器决定即将到来的I/O活动如何被调度。标准调度器如noop或cfq表现得非常合理。

I/O调度器可以在kernel\_source/block中找到。

1. 复制你的I/O调度器文件（例如，sio-iosched.c），然后浏览到kernel\_source/block。将文件粘贴到那里。调度器文件。
2. 现在打开Kconfig.iosched并将你的选择添加到Kconfig中，例如针对SIO：

```
config IOSCHED_SIO
 tristate "简单I/O调度器"
 默认 y
 ---帮助---
简单I/O调度器是一个极其简单的调度器， 基于noop和deadline，依赖截止时间
来 确保公平性。该算法不进行任何排序，只做基本合并，尽量保持最低开销
。它主要针对随机访问设备（例如：闪存设备）。
```

3. 然后按如下设置默认选项：

默认 "sio" 如果 DEFAULT\_SIO

保存文件。

4. 打开kernel\_source/block/中的Makefile，简单地为SIO添加以下行：

obj-\$(CONFIG\_IOSCHED\_SIO) += sio-iosched.o

保存文件，完成！I/O调度器现在应该会出现在菜单配置中。

GitHub 上的类似提交：添加了简单的 I/O 调度器。

## Section 257.3: I/O Schedulers

You can enhance your kernel by adding new I/O schedulers if needed. Globally, governors and schedulers are the same; they both provide a way how the system should work. However, for the schedulers it is all about the input/output datastream except for the CPU settings. I/O schedulers decide how an upcoming I/O activity will be scheduled. The standard schedulers such as *noop* or *cfq* are performing very reasonably.

I/O schedulers can be found in *kernel\_source/block*.

1. Copy your I/O scheduler file (for example, *sio-iosched.c*) and browse to *kernel\_source/block*. Paste the scheduler file there.
2. Now open *Kconfig.iosched* and add your choice to the *Kconfig*, for example for *SIO*:

```
config IOSCHED_SIO
 tristate "Simple I/O scheduler"
 default y
 ---help---
The Simple I/O scheduler is an extremely simple scheduler,
based on noop and deadline, that relies on deadlines to
ensure fairness. The algorithm does not do any sorting but
basic merging, trying to keep a minimum overhead. It is aimed
mainly for aleatory access devices (eg: flash devices).
```

3. Then set the default choice option as follows:

default "sio" if DEFAULT\_SIO

Save the file.

4. Open the *Makefile* in *kernel\_source/block/* and simply add the following line for *SIO*:

obj-\$(CONFIG\_IOSCHED\_SIO) += sio-iosched.o

Save the file and you are done! The I/O schedulers should now pop up at the menu config.

Similar commit on GitHub: [added Simple I/O scheduler](#).

# 第 258 章：内存泄漏

## 第 258.1 节：避免使用 AsyncTask 导致 Activity 泄漏

**一句警告：**AsyncTask 除了这里描述的内存泄漏问题外，还有许多陷阱。所以使用这个 API 时要小心，或者如果你不完全理解其影响，最好完全避免使用。还有许多替代方案（Thread、EventBus、RxAndroid 等）。

使用AsyncTask的一个常见错误是捕获对宿主Activity（或Fragment）的强引用：

```
类 MyActivity extends Activity {
 私有 AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
 // 不要这样做！内部类会隐式持有它们的
 // 父类的指针，在这里就是 Activity！
 }
}
```

这是个问题，因为AsyncTask很容易比父Activity存活得更久，例如当任务运行时发生配置更改。

正确的做法是将你的任务设为一个静态类，这样不会捕获父类，并持有对宿主Activity的弱引用：

```
类 MyActivity extends Activity {
 静态类 MyTask extends AsyncTask<Void, Void, Void> {
 // 弱引用仍然允许 Activity 被垃圾回收
 private final WeakReference<MyActivity> weakActivity;

 MyTask(MyActivity myActivity) {
 this.weakActivity = new WeakReference<>(myActivity);
 }

 @Override
 public Void doInBackground(Void... params) {
 // 在这里执行异步操作
 }

 @Override
 public void onPostExecute(Void result) {
 // 重新获取对 Activity 的强引用，并验证
 // 它仍然存在且处于活动状态。
 MyActivity activity = weakActivity.get();
 if (activity == null
 || activity.isFinishing()
 || activity.isDestroyed()) {
 // Activity 不再有效，什么也不做！
 return;
 }

 // Activity 仍然有效，在这里执行主线程操作
 }
}
```

# Chapter 258: Memory Leaks

## Section 258.1: Avoid leaking Activities with AsyncTask

**A word of caution:** AsyncTask has [many gotcha's](#) apart from the memory leak described here. So be careful with this API, or avoid it altogether if you don't fully understand the implications. There are many alternatives (Thread, EventBus, RxAndroid, etc).

One common mistake with AsyncTask is to capture a strong reference to the host Activity (or Fragment):

```
class MyActivity extends Activity {
 private AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
 // Don't do this! Inner classes implicitly keep a pointer to their
 // parent, which in this case is the Activity!
 }
}
```

This is a problem because AsyncTask can easily outlive the parent Activity, for example if a configuration change happens while the task is running.

The right way to do this is to make your task a **static** class, which does not capture the parent, and holding a [weak reference](#) to the host Activity:

```
class MyActivity extends Activity {
 static class MyTask extends AsyncTask<Void, Void, Void> {
 // Weak references will still allow the Activity to be garbage-collected
 private final WeakReference<MyActivity> weakActivity;

 MyTask(MyActivity myActivity) {
 this.weakActivity = new WeakReference<>(myActivity);
 }

 @Override
 public Void doInBackground(Void... params) {
 // do async stuff here
 }

 @Override
 public void onPostExecute(Void result) {
 // Re-acquire a strong reference to the activity, and verify
 // that it still exists and is active.
 MyActivity activity = weakActivity.get();
 if (activity == null
 || activity.isFinishing()
 || activity.isDestroyed()) {
 // activity is no longer valid, don't do anything!
 return;
 }

 // The activity is still valid, do main-thread stuff here
 }
 }
}
```

## 第258.2节：常见的内存泄漏及其修复方法

### 1. 修正你的上下文：

尝试使用合适的上下文：例如，由于Toast可以在多个活动中显示，而不仅仅是在一个活动中，对Toast使用getApplicationContext()，并且由于服务即使在活动结束后仍可继续运行，启动服务时使用：

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

使用此表作为快速指南，了解何种上下文是合适的：

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO <sup>1</sup>	YES	NO <sup>1</sup>	NO <sup>1</sup>	NO <sup>1</sup>
Layout Inflation	NO <sup>2</sup>	YES	NO <sup>2</sup>	NO <sup>2</sup>	NO <sup>2</sup>
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO <sup>3</sup>
Load Resource Values	YES	YES	YES	YES	YES

[关于上下文的原始文章在这里。](#)

### 2. 对Context的静态引用

一个严重的内存泄漏错误是保持对View的静态引用。每个View都有一个对Context的内部引用。这意味着旧的活动及其整个视图层次结构在应用程序终止之前不会被垃圾回收。旋转屏幕时，你的应用程序会在内存中存在两份。

确保绝对没有对View、Context或它们任何子类的静态引用。

### 3. 检查你是否真的完成了你的服务。

例如，我有一个使用谷歌定位服务API的IntentService。但我忘记调用googleApiClient.disconnect();:

```
//在onDestroy()中断开API连接
if (googleApiClient.isConnected()) {
 LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
 GoogleLocationService.this);
 googleApiClient.disconnect();
}
```

### 4. 检查图片和位图的使用：

如果你使用Square的**Picasso库**，我发现不使用.fit()会导致内存泄漏，这大幅度将我的内存占用从平均50MB减少到不到19MB：

```
Picasso.with(ActivityExample.this) //Activity上下文
 .load(object.getImageUrl())
 .fit() //这避免了OutOfMemoryError
```

## Section 258.2: Common memory leaks and how to fix them

### 1. Fix your contexts:

Try using the appropriate context: For example since a Toast can be seen in many activities instead of just one, use getApplicationContext() for toasts, and since services can keep running even though an activity has ended start a service with:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Use this table as a quick guide for what context is appropriate:

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO <sup>1</sup>	YES	NO <sup>1</sup>	NO <sup>1</sup>	NO <sup>1</sup>
Layout Inflation	NO <sup>2</sup>	YES	NO <sup>2</sup>	NO <sup>2</sup>	NO <sup>2</sup>
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO <sup>3</sup>
Load Resource Values	YES	YES	YES	YES	YES

Original [article on context here.](#)

### 2. Static reference to Context

A serious memory leak mistake is keeping a static reference to View. Every View has an inner reference to the Context. Which means an old Activity with its whole view hierarchy will not be garbage collected until the app is terminated. You will have your app twice in memory when rotating the screen.

Make sure there is absolutely no static reference to View, Context or any of their descendants.

### 3. Check that you're actually finishing your services.

For example, I have an intentService that uses the Google location service API. And I forgot to call googleApiClient.disconnect();:

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
 LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
 GoogleLocationService.this);
 googleApiClient.disconnect();
}
```

### 4. Check image and bitmaps usage:

If you are using Square's **library Picasso** I found I was leaking memory by not using the .fit(), that drastically reduced my memory footprint from 50MB in average to less than 19MB:

```
Picasso.with(ActivityExample.this) //Activity context
 .load(object.getImageUrl())
 .fit() //This avoided the OutOfMemoryError
```

```
.centerCrop() //使图像不被拉伸
.into(imageView);
```

5. 如果你使用广播接收器，请注销它们。
6. 如果你使用java.util.Observer (观察者模式) :

确保使用deleteObserver(observer);

## 第258.3节：使用LeakCanary库检测内存泄漏

LeakCanary是一个开源的Java库，用于在调试版本中检测内存泄漏。

只需在build.gradle中添加依赖：

```
dependencies {
 debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
 releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
 testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
}
```

然后在你的Application类中：

```
public class ExampleApplication extends Application {

 @Override public void onCreate() {
 super.onCreate();

 if (LeakCanary.isInAnalyzerProcess(this)) {
 // 该进程专用于 LeakCanary 的堆分析。
 // 你不应该在此进程中初始化你的应用程序。
 return;
 }

 LeakCanary.install(this);
 }
}
```

现在，当在你的**debug**

注意：发布版本代码中除了 leakcanary-android-no-op 依赖中存在的两个空类外，不会包含任何 LeakCanary 的引用。leakcanary-android-no-op 依赖。

## 第 258.4 节：活动中的匿名回调

每次创建匿名类时，它都会隐式地保留对其父类的引用。所以当你写：

```
public class LeakyActivity extends Activity {

 ...

 foo.registerCallback(new BarCallback()
 {
 @Override
 public void onBar()
 })
```

```
.centerCrop() //makes image to not stretch
.into(imageView);
```

5. If you are using broadcast receivers unregister them.

6. If you are using java.util.Observer (Observer pattern):

Make sure to use deleteObserver(observer);

## Section 258.3: Detect memory leaks with the LeakCanary library

LeakCanary is an Open Source Java library to detect memory leaks in your debug builds.

Just add the dependencies in the build.gradle:

```
dependencies {
 debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
 releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
 testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
}
```

Then in your Application class:

```
public class ExampleApplication extends Application {

 @Override public void onCreate() {
 super.onCreate();

 if (LeakCanary.isInAnalyzerProcess(this)) {
 // This process is dedicated to LeakCanary for heap analysis.
 // You should not init your app in this process.
 return;
 }

 LeakCanary.install(this);
 }
}
```

Now LeakCanary will automatically show a notification when an activity memory leak is detected in your **debug** build.

NOTE: Release code will contain no reference to LeakCanary other than the two empty classes that exist in the leakcanary-android-no-op dependency.

## Section 258.4: Anonymous callback in activities

Every Time you create an anonymous class, it retains an implicit reference to its parent class. So when you write:

```
public class LeakyActivity extends Activity {

 ...

 foo.registerCallback(new BarCallback()
 {
 @Override
 public void onBar()
 })
```

```

 {
 // 执行某些操作
 });
}

```

实际上，你是将你的 LeakyActivity 实例的引用传递给了 foo。当用户离开你的 LeakyActivity 时，这个引用可能会阻止 LeakyActivity 实例被垃圾回收。这是一个严重的内存泄漏，因为活动持有对其整个视图层次结构的引用，因此在内存中是相当大的对象。

如何避免此内存泄漏：

你当然可以完全避免在活动中使用匿名回调。你也可以根据活动的生命周期注销所有回调。示例如下：

```

public class NonLeakyActivity extends Activity
{
 private final BarCallback mBarCallback = new BarCallback()
 {
 @Override
 public void onBar()
 {
 // 执行某些操作
 }
 };

 @Override
 protected void onResume()
 {
 super.onResume();
 foo.registerCallback(mBarCallback);
 }

 @Override
 protected void onPause()
 {
 super.onPause();
 foo.unregisterCallback(mBarCallback);
 }
}

```

## 第258.5节：静态类中的Activity上下文

通常你会想将一些Android类封装成更易用的工具类。这些工具类通常需要一个上下文来访问Android操作系统或你的应用资源。一个常见的例子是对 SharedPreferences类的封装。为了访问Android的共享偏好设置，必须写：

```
context.getSharedPreferences(prefName, mode);
```

因此，有人可能会想创建如下类：

```

public class LeakySharedPrefsWrapper
{
 private static Context sContext;

 public static void init(Context context)
 {
 sContext = context;
 }
}

```

```

 {
 // do something
 });
}

```

You are in fact sending a reference to your LeakyActivity instance to foo. When the user navigates away from your LeakyActivity, this reference can prevent the LeakyActivity instance from being garbage collected. This is a serious leak as activities hold a reference to their entire view hierarchy and are therefore rather large objects in memory.

How to avoid this leak:

You can of course avoid using anonymous callbacks in activities entirely. You can also unregister all of your callbacks with respect to the activity lifecycle. like so:

```

public class NonLeakyActivity extends Activity
{
 private final BarCallback mBarCallback = new BarCallback()
 {
 @Override
 public void onBar()
 {
 // do something
 }
 };

 @Override
 protected void onResume()
 {
 super.onResume();
 foo.registerCallback(mBarCallback);
 }

 @Override
 protected void onPause()
 {
 super.onPause();
 foo.unregisterCallback(mBarCallback);
 }
}

```

## Section 258.5: Activity Context in static classes

Often you will want to wrap some of Android's classes in easier to use utility classes. Those utility classes often require a context to access the android OS or your apps' resources. A common example of this is a wrapper for the SharedPreferences class. In order to access Androids shared preferences one must write:

```
context.getSharedPreferences(prefName, mode);
```

And so one may be tempted to create the following class:

```

public class LeakySharedPrefsWrapper
{
 private static Context sContext;

 public static void init(Context context)
 {
 sContext = context;
 }
}

```

```

public int getInt(String name, int defValue)
{
 return sContext.getSharedPreferences("a_name", Context.MODE_PRIVATE).getInt(name, defValue);
}

```

现在，如果你用你的活动上下文调用init()，LeakySharedPrefsWrapper将会保留对你的活动的引用，导致它无法被垃圾回收。

如何避免：

调用静态辅助函数时，可以传入应用程序上下文，使用  
context.getApplicationContext();

创建静态辅助函数时，可以从你获得的上下文中提取应用程序上下文  
(在应用程序上下文中调用getApplicationContext()会返回应用程序上下文)。所以我们  
的包装器的修复方法很简单：

```

public static void init(Context context)
{
 sContext = context.getApplicationContext();
}

```

如果应用程序上下文不适合您的使用场景，您可以在每个工具函数中包含一个 Context 参数，但应避免持有这些 Context 参数的引用。在这种情况下，解决方案如下：

```

public int getInt(Context context, String name, int defValue)
{
 // 不要持有 context 的引用以避免潜在的内存泄漏。
 return context.getSharedPreferences("a_name", Context.MODE_PRIVATE).getInt(name, defValue);
}

```

## 第 258.6 节：避免因监听器导致 Activity 泄漏

如果您在 Activity 中实现或创建监听器，请始终关注注册了监听器的对象的生命周期。

考虑一个应用程序，其中有多个不同的 Activity/Fragment 关注用户登录或登出状态。实现方式之一是拥有一个单例的 UserController 实例，订阅该实例以便在用户状态变化时收到通知：

```

public class UserController {
 private static UserController instance;
 private List<StateListener> listeners;

 public static synchronized UserController getInstance() {
 if (instance == null) {
 instance = new UserController();
 }
 return instance;
 }

 private UserController() {
 // 初始化
 }
}

```

```

public int getInt(String name, int defValue)
{
 return sContext.getSharedPreferences("a_name", Context.MODE_PRIVATE).getInt(name, defValue);
}

```

now, if you call init() with your activity context, the LeakySharedPrefsWrapper will retain a reference to your activity, preventing it from being garbage collected.

How to avoid:

When calling static helper functions, you can send in the application context using  
context.getApplicationContext();

When creating static helper functions, you can extract the application context from the context you are given  
(Calling getApplicationContext() on the application context returns the application context). So the fix to our  
wrapper is simple:

```

public static void init(Context context)
{
 sContext = context.getApplicationContext();
}

```

If the application context is not appropriate for your use case, you can include a Context parameter in each utility  
function, you should avoid keeping references to these context parameters. In this case the solution would look like  
so:

```

public int getInt(Context context, String name, int defValue)
{
 // do not keep a reference of context to avoid potential leaks.
 return context.getSharedPreferences("a_name", Context.MODE_PRIVATE).getInt(name, defValue);
}

```

## Section 258.6: Avoid leaking Activities with Listeners

If you implement or create a listener in an Activity, always pay attention to the lifecycle of the object that has the  
listener registered.

Consider an application where we have several different activities/fragments interested in when a user is logged in  
or out. One way of doing this would be to have a singleton instance of a UserController that can be subscribed to  
in order to get notified when the state of the user changes:

```

public class UserController {
 private static UserController instance;
 private List<StateListener> listeners;

 public static synchronized UserController getInstance() {
 if (instance == null) {
 instance = new UserController();
 }
 return instance;
 }

 private UserController() {
 // Init
 }
}

```

```

public void registerUserStateChangeListener(StateListener listener) {
 listeners.add(listener);
}

public void logout() {
 for (StateListener listener : listeners) {
 listener.userLoggedOut();
 }
}

public void login() {
 for (StateListener listener : listeners) {
 listener.userLoggedIn();
 }
}

public interface StateListener {
 void userLoggedIn();
 void userLoggedOut();
}
}

```

然后有两个活动，SignInActivity：

```

public class SignInActivity extends Activity implements UserController.StateListener{

 UserController userController;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 this.userController = UserController.getInstance();
 this.userController.registerUserStateChangeListener(this);
 }

 @Override
 public void userLoggedIn() {
 startMainActivity();
 }

 @Override
 public void userLoggedOut() {
 showLoginForm();
 }

 ...

 public void onLoginClicked(View v) {
 userController.login();
 }
}

```

还有MainActivity：

```

public class MainActivity extends Activity implements UserController.StateListener{
 UserController userController;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

```

```

public void registerUserStateChangeListener(StateListener listener) {
 listeners.add(listener);
}

public void logout() {
 for (StateListener listener : listeners) {
 listener.userLoggedOut();
 }
}

public void login() {
 for (StateListener listener : listeners) {
 listener.userLoggedIn();
 }
}

public interface StateListener {
 void userLoggedIn();
 void userLoggedOut();
}
}

```

Then there are two activities, SignInActivity:

```

public class SignInActivity extends Activity implements UserController.StateListener{

 UserController userController;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 this.userController = UserController.getInstance();
 this.userController.registerUserStateChangeListener(this);
 }

 @Override
 public void userLoggedIn() {
 startMainActivity();
 }

 @Override
 public void userLoggedOut() {
 showLoginForm();
 }

 ...

 public void onLoginClicked(View v) {
 userController.login();
 }
}

```

And MainActivity:

```

public class MainActivity extends Activity implements UserController.StateListener{
 UserController userController;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

```

```

this.userController = UserController.getInstance();
this.userController.registerUserStateChangeListener(this);
}

@Override
public void userLoggedIn() {
 showUserAccount();
}

@Override
public void userLoggedOut() {
 finish();
}

...
public void onLogoutClicked(View v) {
 userController.logout();
}
}

```

这个例子中发生的情况是，每次用户登录然后再次登出时，都会泄漏一个MainActivity实例。泄漏发生的原因是UserController#listeners中存在对该活动的引用。

请注意：即使我们使用匿名内部类作为监听器，活动仍然会泄漏：

```

...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
 @Override
 public void userLoggedIn() {
 showUserAccount();
 }

 @Override
 public void userLoggedOut() {
 finish();
 }
});
...

```

活动仍然会泄漏，因为匿名内部类对外部类（在本例中是活动）有隐式引用。这就是为什么可以从内部类调用外部类的实例方法。实际上，唯一不持有外部类引用的内部类类型是static内部类。

简而言之，所有非静态内部类的实例都持有创建它们的外部类实例的隐式引用。

解决此问题的主要方法有两种，一种是添加方法以移除监听器，另一种是UserController#listeners或使用WeakReference来持有监听器的引用。

### 方案一：移除监听器

让我们先创建一个新方法removeUserStateChangeListener(StateListener listener)：

```

public class UserController {

 ...
 public void registerUserStateChangeListener(StateListener listener) {

```

```

 this.userController = UserController.getInstance();
 this.userController.registerUserStateChangeListener(this);
 }

 @Override
 public void userLoggedIn() {
 showUserAccount();
 }

 @Override
 public void userLoggedOut() {
 finish();
 }

 ...
 public void onLogoutClicked(View v) {
 userController.logout();
 }
}

```

What happens with this example is that every time the user logs in and then logs out again, a MainActivity instance is leaked. The leak occurs because there is a reference to the activity in UserController#listeners.

**Please note:** Even if we use an anonymous inner class as a listener, the activity would still leak:

```

...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
 @Override
 public void userLoggedIn() {
 showUserAccount();
 }

 @Override
 public void userLoggedOut() {
 finish();
 }
});
...

```

The activity would still leak, because the anonymous inner class has an implicit reference to the outer class (in this case the activity). This is why it is possible to call instance methods in the outer class from the inner class. In fact, the only type of inner classes that do not have a reference to the outer class are **static** inner classes.

In short, all instances of non-static inner classes hold an implicit reference to the instance of the outer class that created them.

There are two main approaches to solving this, either by adding a method to remove a listener from UserController#listeners or using a [WeakReference](#) to hold the reference of the listeners.

### Alternative 1: Removing listeners

Let us start by creating a new method removeUserStateChangeListener(StateListener listener):

```

public class UserController {

 ...
 public void registerUserStateChangeListener(StateListener listener) {

```

```

 listeners.add(listener);
 }

 public void removeUserStateChangeListener(StateListener listener) {
 listeners.remove(listener);
 }

 ...
}

```

然后让我们在活动的onDestroy方法中调用此方法：

```

public class MainActivity extends Activity implements UserController.StateListener{
 ...

 @Override
 protected void onDestroy() {
 super.onDestroy();
 userController.removeUserStateChangeListener(this);
 }
}

```

通过此修改，当用户登录和注销时，MainActivity 的实例不再泄漏。然而，如果文档不够清晰，下一位开始使用UserController的开发者很可能会忽略在活动销毁时注销监听器的必要性，这就引出了避免此类泄漏的第二种方法。

## 替代方案2：使用弱引用

首先，让我们解释一下什么是弱引用。顾名思义，弱引用是对对象的弱引用。与作为强引用的普通实例字段相比，弱引用不会阻止垃圾回收器（GC）回收对象。在上面的例子中，如果UserController使用WeakReference来引用监听器，那么在MainActivity被销毁后，它就可以被垃圾回收。

简而言之，弱引用告诉GC，如果没有其他强引用指向该对象，就可以将其移除。

让我们修改UserController，使用WeakReference列表来跟踪它的监听器：

```

public class UserController {

 ...

 private List<WeakReference<StateListener>> listeners;
 ...

 public void registerUserStateChangeListener(StateListener listener) {
 listeners.add(new WeakReference<>(listener));
 }

 public void removeUserStateChangeListener(StateListener listenerToRemove) {
 WeakReference referencesToRemove = null;
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null && listener == listenerToRemove) {
 referencesToRemove = listenerRef;
 break;
 }
 }
 }
}

```

```

 listeners.add(listener);
 }

 public void removeUserStateChangeListener(StateListener listener) {
 listeners.remove(listener);
 }

 ...
}

```

Then let us call this method in the activity's onDestroy method:

```

public class MainActivity extends Activity implements UserController.StateListener{
 ...

 @Override
 protected void onDestroy() {
 super.onDestroy();
 userController.removeUserStateChangeListener(this);
 }
}

```

With this modification the instances of MainActivity are no longer leaked when the user logs in and out. However, if the documentation isn't clear, chances are that the next developer that starts using UserController might miss that it is required to unregister the listener when the activity is destroyed, which leads us to the second method of avoiding these types of leaks.

## Alternative 2: Using weak references

First off, let us start by explaining what a weak reference is. A weak reference, as the name suggests, holds a weak reference to an object. Compared to a normal instance field, which is a strong reference, a weak references does not stop the garbage collector, GC, from removing the objects. In the example above this would allow MainActivity to be garbage-collected after it has been destroyed if the UserController used [WeakReference](#) to the reference the listeners.

In short, a weak reference is telling the GC that if no one else has a strong reference to this object, go ahead and remove it.

Let us modify the UserController to use a list of [WeakReference](#) to keep track of its listeners:

```

public class UserController {

 ...

 private List<WeakReference<StateListener>> listeners;
 ...

 public void registerUserStateChangeListener(StateListener listener) {
 listeners.add(new WeakReference<>(listener));
 }

 public void removeUserStateChangeListener(StateListener listenerToRemove) {
 WeakReference referencesToRemove = null;
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null && listener == listenerToRemove) {
 referencesToRemove = listenerRef;
 break;
 }
 }
 }
}

```

```

 }
 listeners.remove(referencesToRemove);
 }

 public void logout() {
 List referencesToRemove = new LinkedList();
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null) {
 listener.userLoggedOut();
 } else {
 referencesToRemove.add(listenerRef);
 }
 }
 }

 public void login() {
 List referencesToRemove = new LinkedList();
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null) {
 listener.userLoggedIn();
 } else {
 referencesToRemove.add(listenerRef);
 }
 }
 }

 ...
}

```

通过此修改，无论监听器是否被移除都无关紧要，因为UserController不持有对任何监听器的强引用。然而，每次编写这些样板代码都很繁琐。

因此，让我们创建一个名为WeakCollection的通用类：

```

public class WeakCollection<T> {
 private LinkedList<WeakReference<T>> list;

 public WeakCollection() {
 this.list = new LinkedList<>();
 }

 public void put(T item){
 //确保如果我们已经有该引用，则不重新添加该项。
 List<T> currentList = get();
 for(T oldItem : currentList){
 if(item == oldItem){
 return;
 }
 }
 list.add(new WeakReference<T>(item));
 }

 public List<T> get() {
 List<T> ret = new ArrayList<>(list.size());
 List<WeakReference<T>> itemsToRemove = new LinkedList<>();
 for (WeakReference<T> ref : list) {
 T item = ref.get();
 if (item == null) {
 itemsToRemove.add(ref);
 } else {
 ret.add(item);
 }
 }
 }
}

```

```

 }
 listeners.remove(referencesToRemove);
 }

 public void logout() {
 List referencesToRemove = new LinkedList();
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null) {
 listener.userLoggedOut();
 } else {
 referencesToRemove.add(listenerRef);
 }
 }
 }

 public void login() {
 List referencesToRemove = new LinkedList();
 for (WeakReference<StateListener> listenerRef : listeners) {
 StateListener listener = listenerRef.get();
 if (listener != null) {
 listener.userLoggedIn();
 } else {
 referencesToRemove.add(listenerRef);
 }
 }
 }

 ...
}

```

With this modification it doesn't matter whether or not the listeners are removed, since UserController holds no strong references to any of the listeners. However, writing this boilerplate code every time is cumbersome. Therefore, let us create a generic class called WeakCollection:

```

public class WeakCollection<T> {
 private LinkedList<WeakReference<T>> list;

 public WeakCollection() {
 this.list = new LinkedList<>();
 }

 public void put(T item){
 //Make sure that we don't re add an item if we already have the reference.
 List<T> currentList = get();
 for(T oldItem : currentList){
 if(item == oldItem){
 return;
 }
 }
 list.add(new WeakReference<T>(item));
 }

 public List<T> get() {
 List<T> ret = new ArrayList<>(list.size());
 List<WeakReference<T>> itemsToRemove = new LinkedList<>();
 for (WeakReference<T> ref : list) {
 T item = ref.get();
 if (item == null) {
 itemsToRemove.add(ref);
 } else {
 ret.add(item);
 }
 }
 }
}

```

```

 }
 for (WeakReference ref : itemsToRemove) {
 this.list.remove(ref);
 }
 return ret;
 }

 public void remove(T listener) {
 WeakReference<T> refToRemove = null;
 for (WeakReference<T> ref : list) {
 T item = ref.get();
 if (item == listener) {
 refToRemove = ref;
 }
 }
 if (refToRemove != null){
 list.remove(refToRemove);
 }
 }
}

```

现在让我们重写UserController以使用WeakCollection<T>：

```

public class UserController {
 ...
 private WeakCollection<StateListener> listenerRefs;
 ...

 public void registerUserStateChangeListener(StateListener listener) {
 listenerRefs.put(listener);
 }

 public void removeUserStateChangeListener(StateListener listenerToRemove) {
 listenerRefs.remove(listenerToRemove);
 }

 public void logout() {
 for (StateListener listener : listenerRefs.get()) {
 listener.userLoggedOut();
 }
 }

 public void login() {
 for (StateListener listener : listenerRefs.get()) {
 listener.userLoggedIn();
 }
 }
 ...
}

```

如上面代码示例所示，WeakCollection<T>消除了使用WeakReference代替普通列表所需的所有样板代码。最重要的是：如果错过调用UserController#removeUserStateChangeListener(StateListener)，监听器及其引用的所有对象都不会发生内存泄漏。

## 第258.7节：避免使用匿名类、

```

 }
 for (WeakReference ref : itemsToRemove) {
 this.list.remove(ref);
 }
 return ret;
 }

 public void remove(T listener) {
 WeakReference<T> refToRemove = null;
 for (WeakReference<T> ref : list) {
 T item = ref.get();
 if (item == listener) {
 refToRemove = ref;
 }
 }
 if (refToRemove != null){
 list.remove(refToRemove);
 }
 }
}

```

Now let us re-write UserController to use WeakCollection<T> instead:

```

public class UserController {
 ...
 private WeakCollection<StateListener> listenerRefs;
 ...

 public void registerUserStateChangeListener(StateListener listener) {
 listenerRefs.put(listener);
 }

 public void removeUserStateChangeListener(StateListener listenerToRemove) {
 listenerRefs.remove(listenerToRemove);
 }

 public void logout() {
 for (StateListener listener : listenerRefs.get()) {
 listener.userLoggedOut();
 }
 }

 public void login() {
 for (StateListener listener : listenerRefs.get()) {
 listener.userLoggedIn();
 }
 }
 ...
}

```

As shown in the code example above, the WeakCollection<T> removes all of the boilerplate code needed to use WeakReference instead of a normal list. To top it all off: If a call to UserController#removeUserStateChangeListener(StateListener) is missed, the listener, and all the objects it is referencing, will not leak.

## Section 258.7: Avoid memory leaks with Anonymous Class,

## Handler、Timer任务、线程导致的内存泄漏

在Android中，每个开发者至少在一个项目中使用过匿名类（Runnable）。任何匿名类都会引用其父类（Activity）。如果我们执行一个长时间运行的任务，父Activity在任务结束之前不会被销毁。

示例使用了Handler和匿名Runnable类。当我们在Runnable未完成时退出Activity，内存将会泄漏。

```
new Handler().postDelayed(new Runnable() {
 @Override
 public void run() {
 // 执行 abc, 持续约 5 秒
 }
}, 10000); // 10 秒后运行 "do abc"。这与定时器、线程相同...
```

我们该如何解决？

1. 不要用匿名类执行任何长时间操作，或者需要一个静态类来处理，并传入弱引用（例如 activity、view...）。线程与匿名类相同。
2. 在 activity 销毁时取消Handler、Timer。

## Handler, Timer Task, Thread

In android, every developer uses Anonymous Class (Runnable) at least once in a project. Any Anonymous Class has a reference to its parent (activity). If we perform a long-running task, the parent activity will not be destroyed until the task is ended.

Example uses handler and Anonymous Runnable class. The memory will be leak when we quit the activity before the Runnable is finished.

```
new Handler().postDelayed(new Runnable() {
 @Override
 public void run() {
 // do abc long 5s or so
 }
}, 10000); // run "do abc" after 10s. It same as timer, thread...
```

How do we solve it?

1. Don't do any long operating with Anonymous Class or we need a Static class for it and pass WeakReference into it (such as activity, view...). Thread is the same with Anonymous Class.
2. Cancel the Handler, Timer when activity is destroyed.

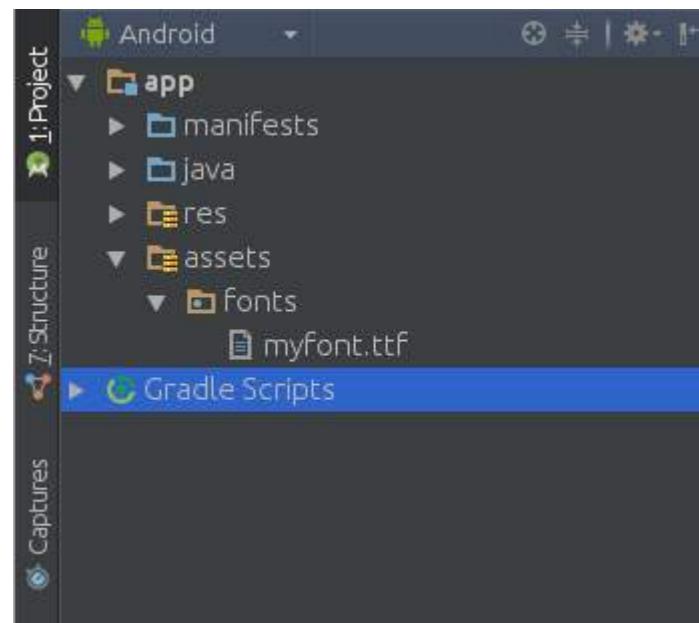
# 第 259 章：使用图标字体提升 Android 性能

## 第 259.1 节：如何集成图标字体

为了使用图标字体，只需按照以下步骤操作：

- 将字体文件添加到您的项目中

您可以通过在线网站如icomoon创建字体图标文件，在那里您可以上传所需图标的SVG文件，然后下载生成的图标字体。接着，将.ttf字体文件放入assets文件夹中名为fonts（您可以自定义名称）的文件夹中：



- 创建辅助类

现在，创建以下辅助类，以避免重复编写字体初始化代码：

```
public class FontManager {
 public static final String ROOT = "fonts/";
 FONT_AWESOME = ROOT + "myfont.ttf";
 public static Typeface getTypeface(Context context) {
 return Typeface.createFromAsset(context.getAssets(), FONT_AWesome);
 }
}
```

您可以使用Typeface类从assets中选择字体。这样，您可以将字体应用到各种视图，例如按钮：

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

现在，按钮的字体已更改为新创建的图标字体。

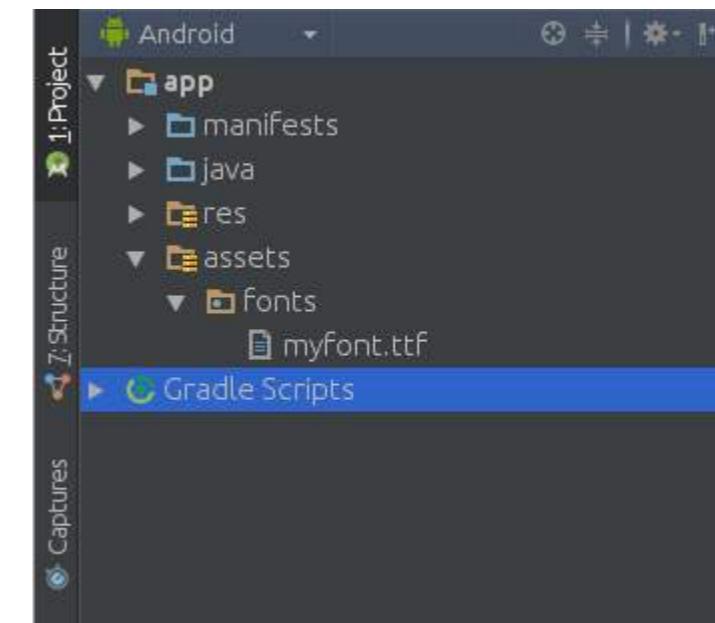
# Chapter 259: Enhancing Android Performance Using Icon Fonts

## Section 259.1: How to integrate icon fonts

In order to use icon fonts, just follow the steps below:

- Add the font file to your project

You may create your font icon file from online websites such as [icomoon](#), where you can upload SVG files of the required icons and then download the created icon font. Then, place the .ttf font file into a folder named fonts (name it as you wish) in the assets folder:



- Create a Helper Class

Now, create the following helper class, so that you can avoid repeating the initialisation code for the font:

```
public class FontManager {
 public static final String ROOT = "fonts/";
 FONT_AWESOME = ROOT + "myfont.ttf";
 public static Typeface getTypeface(Context context) {
 return Typeface.createFromAsset(context.getAssets(), FONT_AWesome);
 }
}
```

You may use the Typeface class in order to pick the font from the assets. This way you can set the typeface to various views, for example, to a button:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

Now, the button typeface has been changed to the newly created icon font.

- 选择您想要的图标

打开附加到图标字体的styles.css文件。在那里你会找到带有你图标Unicode字符的样式：

```
.icon-arrow-circle-down:before {
 content: "\e001";
}
.icon-arrow-circle-left:before {
 content: "\e002";
}
.icon-arrow-circle-o-down:before {
 content: "\e003";
}
.icon-arrow-circle-o-left:before {
 content: "\e004";
}
```

该资源文件将作为字典，映射与特定图标关联的Unicode字符到可读名称。现在，按如下方式创建字符串资源：

```
<resources>
 <! — 图标字体 -->
 <string name="icon_arrow_circle_down">#xe001; </string>
 <string name="icon_arrow_circle_left">#xe002; </string>
 <string name="icon_arrow_circle_o_down">#xe003; </string>
 <string name="icon_arrow_circle_o_left">#xe004; </string>
</resources>
```

- 在代码中使用图标

现在，您可以在各种视图中使用您的字体，例如：

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

您也可以使用图标字体创建按钮文本视图：

- Pick up the icons you want

Open the *styles.css* file attached to the icon font. There you will find the styles with Unicode characters of your icons:

```
.icon-arrow-circle-down:before {
 content: "\e001";
}
.icon-arrow-circle-left:before {
 content: "\e002";
}
.icon-arrow-circle-o-down:before {
 content: "\e003";
}
.icon-arrow-circle-o-left:before {
 content: "\e004";
}
```

This resource file will serve as a dictionary, which maps the Unicode character associated with a specific icon to a human-readable name. Now, create the string resources as follows:

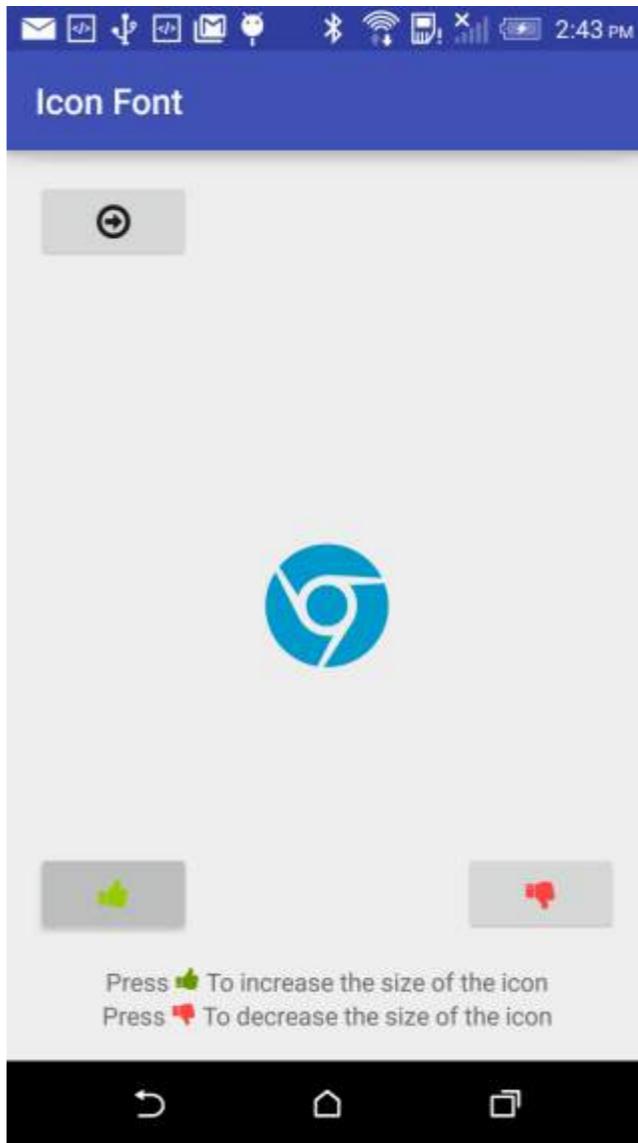
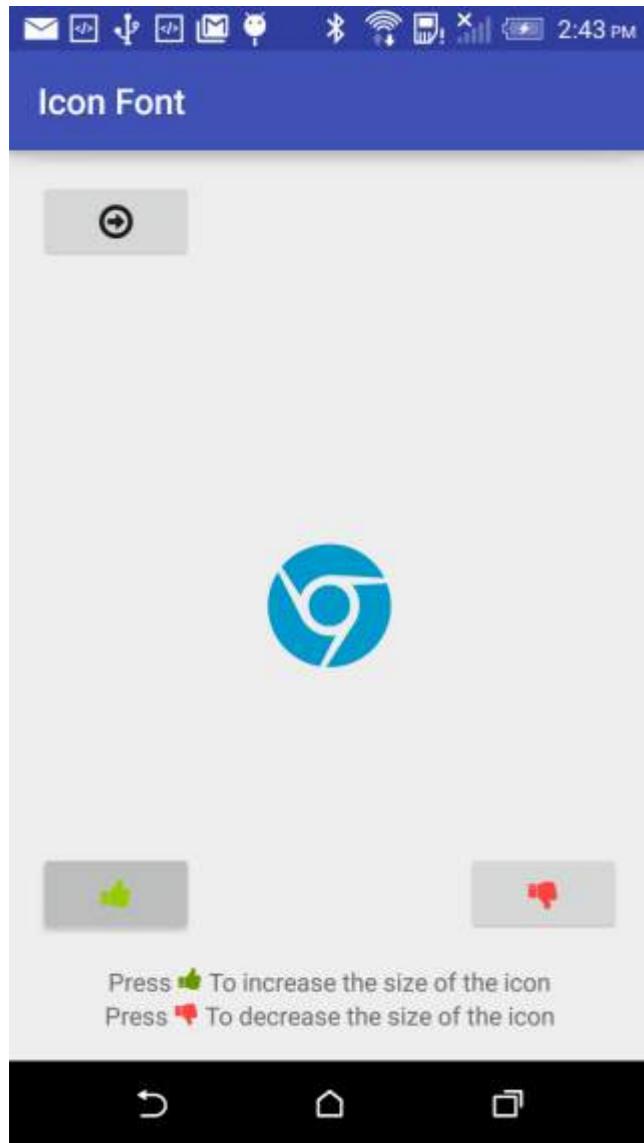
```
<resources>
 <! — Icon Fonts -->
 <string name="icon_arrow_circle_down">#xe001; </string>
 <string name="icon_arrow_circle_left">#xe002; </string>
 <string name="icon_arrow_circle_o_down">#xe003; </string>
 <string name="icon_arrow_circle_o_left">#xe004; </string>
</resources>
```

- Use the icons in your code

Now, you may use your font in various views, for example, as follows:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

You may also create button text views using icon fonts:



## 第259.2节：带图标字体的TabLayout

```
public class TabAdapter extends FragmentPagerAdapter {

 CustomTypefaceSpan fonte;
 List<Fragment> fragments = new ArrayList<>(4);
 private String[] icons = {"\ue001", "\uE002", "\uE003", "\uE004"};

 public TabAdapter(FragmentManager fm, CustomTypefaceSpan fonte) {
 super(fm);
 this.fonte = fonte
 for (int i = 0; i < 4; i++){
 fragments.add(MyFragment.newInstance());
 }
 }

 public List<Fragment> getFragments() {
 return fragments;
 }

 @Override
 public Fragment getItem(int position) {
 return fragments.get(position);
 }
}
```

## Section 259.2: TabLayout with icon fonts

```
public class TabAdapter extends FragmentPagerAdapter {

 CustomTypefaceSpan fonte;
 List<Fragment> fragments = new ArrayList<>(4);
 private String[] icons = {"\ue001", "\uE002", "\uE003", "\uE004"};

 public TabAdapter(FragmentManager fm, CustomTypefaceSpan fonte) {
 super(fm);
 this.fonte = fonte
 for (int i = 0; i < 4; i++){
 fragments.add(MyFragment.newInstance());
 }
 }

 public List<Fragment> getFragments() {
 return fragments;
 }

 @Override
 public Fragment getItem(int position) {
 return fragments.get(position);
 }
}
```

```

@Override
public CharSequence getPageTitle(int position) {
 SpannableStringBuilder ss = new SpannableStringBuilder(Icons[position]);
 ss.setSpan(fonte, 0, ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
 ss.setSpan(new RelativeSizeSpan(1.5f), 0, ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
 return ss;
}

```

```

@Override
public int getCount() {
 return 4;
}

}

```

- 在此示例中，myfont.ttf 位于 Assets 文件夹中。创建 Assets 文件夹
- 在您的活动类中

```

//..
TabLayout tabs;
ViewPager tabs_pager;
public CustomTypefaceSpan fonte;
//..

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 //...
fm = getSupportFragmentManager();
 fonte = new CustomTypefaceSpan("icomoon", Typeface.createFromAsset(getAssets(),"myfont.ttf"));
this.tabs = ((TabLayout) findViewById(R.id.tabs));
 this.tabs_pager = ((ViewPager) findViewById(R.id.tabs_pager));
 //...
}

@Override
protected void onStart() {
 super.onStart();
 //..
tabs_pager.setAdapter(new TabAdapter(fm, fonte));
 tabs.setupWithViewPager(tabs_pager);
 //..
}

```

```

@Override
public CharSequence getPageTitle(int position) {
 SpannableStringBuilder ss = new SpannableStringBuilder(Icons[position]);
 ss.setSpan(fonte, 0, ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
 ss.setSpan(new RelativeSizeSpan(1.5f), 0, ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
 return ss;
}

```

```

@Override
public int getCount() {
 return 4;
}

}

```

- In this example, myfont.ttf is in Assets folder. Creating Assets folder
- In your activity class

```

//..
TabLayout tabs;
ViewPager tabs_pager;
public CustomTypefaceSpan fonte;
//..

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 //...
fm = getSupportFragmentManager();
 fonte = new CustomTypefaceSpan("icomoon", Typeface.createFromAsset(getAssets(),"myfont.ttf"));
this.tabs = ((TabLayout) findViewById(R.id.tabs));
 this.tabs_pager = ((ViewPager) findViewById(R.id.tabs_pager));
 //...
}

@Override
protected void onStart() {
 super.onStart();
 //..
tabs_pager.setAdapter(new TabAdapter(fm, fonte));
 tabs.setupWithViewPager(tabs_pager);
 //..
}

```

# 第260章：位图缓存

参数	详细信息
键	用于在内存缓存中存储位图的键
位图	将缓存到内存中的位图值

内存高效的位图缓存：如果您的应用使用动画，这一点尤其重要，因为动画会在垃圾回收（GC）清理期间停止，导致应用在用户看来变得迟缓。缓存允许重用创建成本高的对象。如果您将对象加载到内存中，可以将其视为该对象的缓存。在Android中处理位图比较棘手。如果您打算重复使用位图，缓存位图就更为重要。

## 第260.1节：使用LRU缓存的位图缓存

### LRU缓存

下面的示例代码演示了使用LruCache类缓存图像的可能实现。

```
private LruCache<String, Bitmap> mMemoryCache;
```

这里的字符串值是位图值的键。

```
// 获取最大可用的虚拟机内存，超过此值将抛出
// OutOfMemory异常。以千字节为单位存储，因为LruCache的构造函数需要一个
// int类型的参数。
final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);

// 使用可用内存的1/8作为此内存缓存的大小。
final int cacheSize = maxMemory / 8;

mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
 @Override
 protected int sizeOf(String key, Bitmap bitmap) {
 // 缓存大小将以千字节为单位测量，而不是
 // 项目数量。
 return bitmap.getByteCount() / 1024;
 }
};
```

### 用于将位图添加到内存缓存

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
 if (getBitmapFromMemCache(key) == null) {
 mMemoryCache.put(key, bitmap);
 }
}
```

### 用于从内存缓存获取位图

```
public Bitmap getBitmapFromMemCache(String key) {
 return mMemoryCache.get(key);
}
```

要将位图加载到ImageView中，只需使用getBitmapFromMemCache("传入键")。

# Chapter 260: Bitmap Cache

Parameter	Details
key	key to store bitmap in memory cache
bitmap	bitmap value which will cache into memory

Memory efficient bitmap caching: This is particularly important if your application uses animations as they will be stopped during GC cleanup and make your application appear sluggish to the user. A cache allows reusing objects which are expensive to create. If you load an object into memory, you can think of this as a cache for the object. Working with bitmap in android is tricky. It is more important to cache the bitmap if you are going to use it repeatedly.

## Section 260.1: Bitmap Cache Using LRU Cache

### LRU Cache

The following example code demonstrates a possible implementation of the LruCache class for caching images.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Here string value is key for bitmap value.

```
// Get max available VM memory, exceeding this amount will throw an
// OutOfMemory exception. Stored in kilobytes as LruCache takes an
// int in its constructor.
final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);

// Use 1/8th of the available memory for this memory cache.
final int cacheSize = maxMemory / 8;

mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
 @Override
 protected int sizeOf(String key, Bitmap bitmap) {
 // The cache size will be measured in kilobytes rather than
 // number of items.
 return bitmap.getByteCount() / 1024;
 }
};
```

### For add bitmap to the memory cache

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
 if (getBitmapFromMemCache(key) == null) {
 mMemoryCache.put(key, bitmap);
 }
}
```

### For get bitmap from memory cache

```
public Bitmap getBitmapFromMemCache(String key) {
 return mMemoryCache.get(key);
}
```

For loading bitmap into ImageView just use **getBitmapFromMemCache("Pass key")**.

# 第261章：有效加载位图

本主题主要集中在如何在安卓设备上有效加载位图。

当涉及加载位图时，问题在于它是从哪里加载的。这里我们将讨论如何从Android设备内的资源加载位图，即例如从图库中加载。

我们将通过下面讨论的示例来讲解。

## 第261.1节：使用Intent从Android设备资源加载图像

使用Intent从图库加载图像。

- 首先你需要有权限

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

- 使用以下代码来设计布局，如下所示。

# Chapter 261: Loading Bitmaps Effectively

This Topic Mainly Concentrate on Loading the Bitmaps Effectively in Android Devices.

When it comes to loading a bitmap, the question comes where it is loaded from. Here we are going to discuss about how to load the Bitmap from Resource with in the Android Device. i.e. eg from Gallery.

We will go through this by example which are discussed below.

## Section 261.1: Load the Image from Resource from Android Device. Using Intents

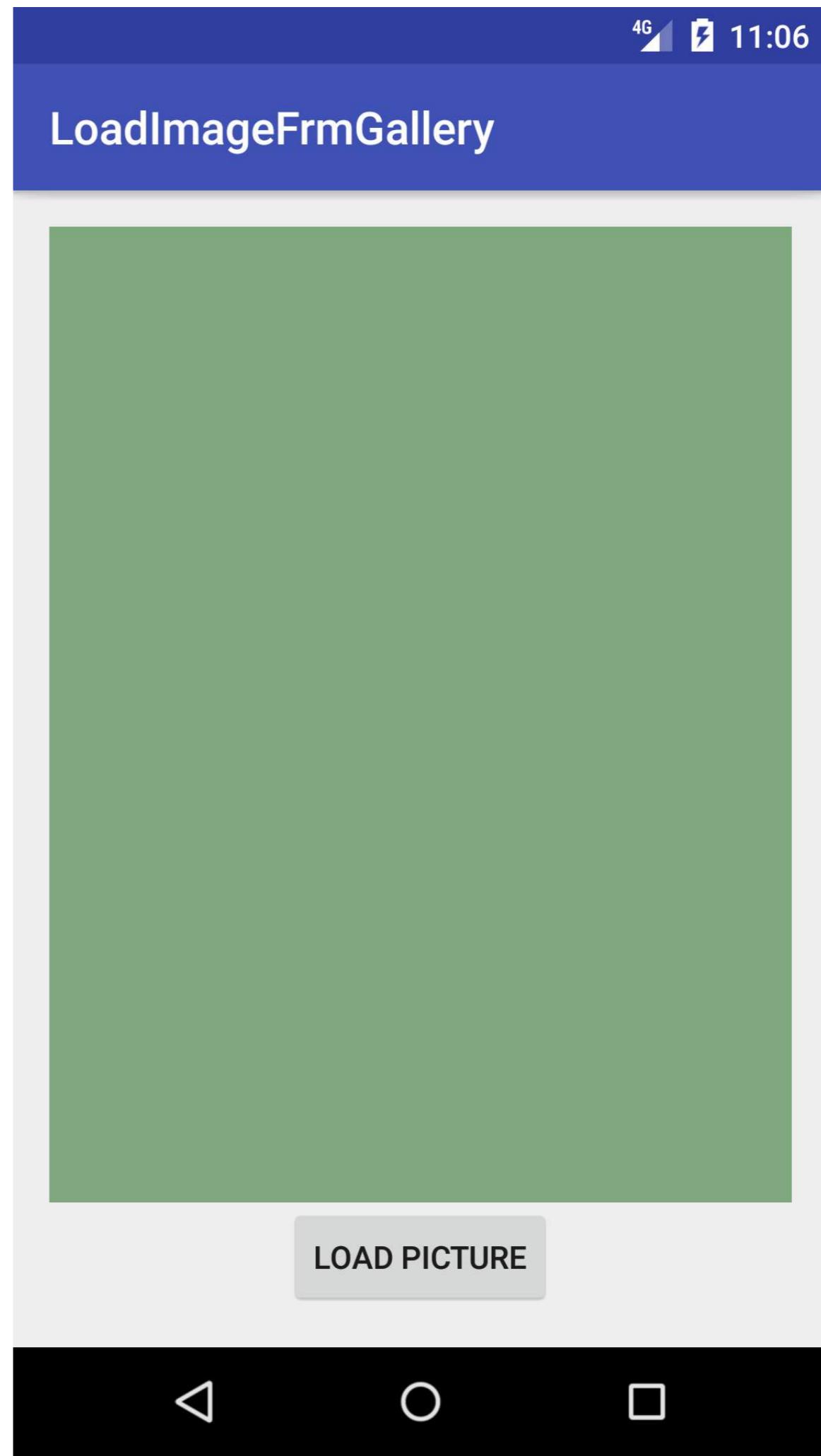
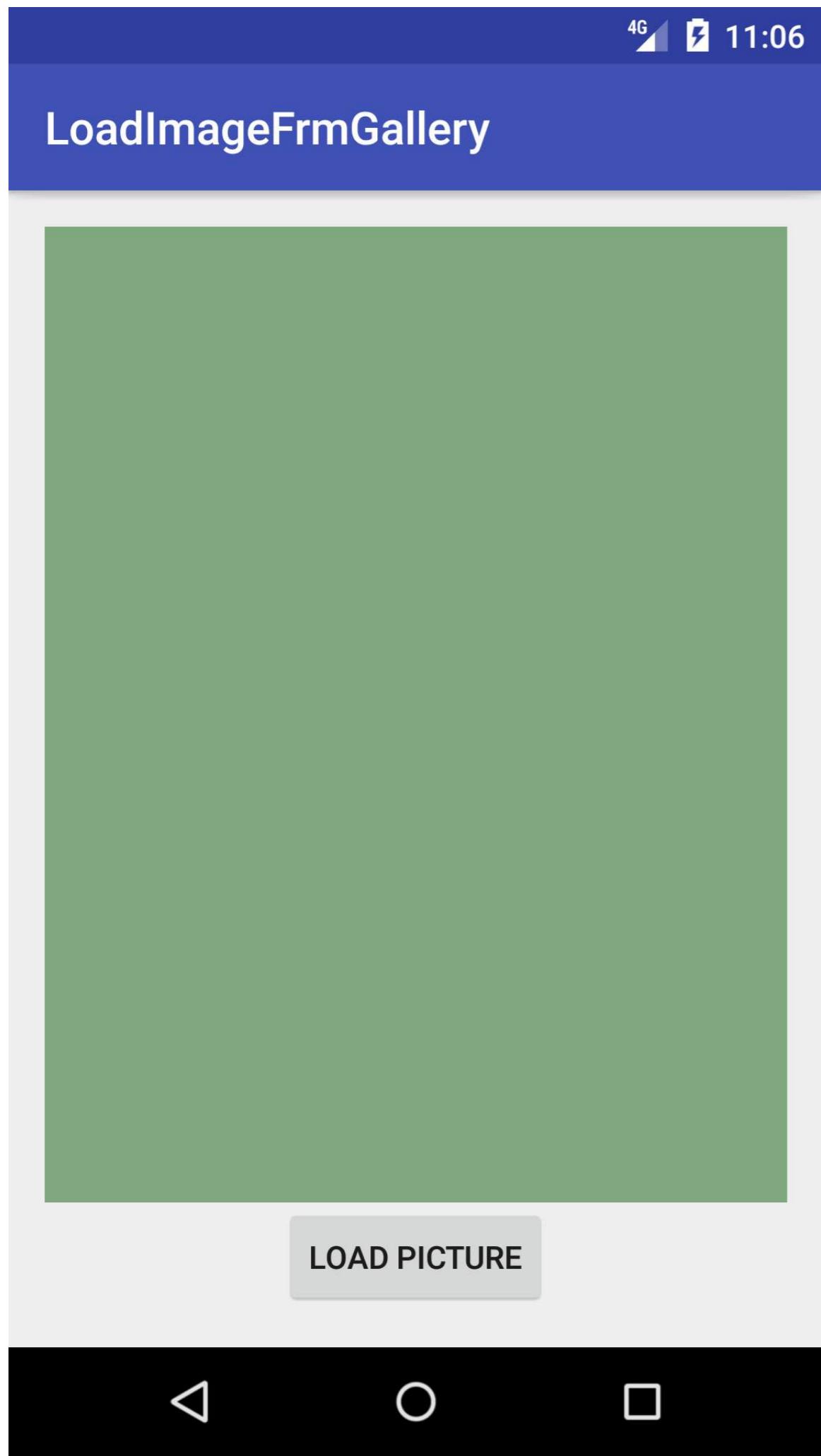
**Using Intents to Load the Image from Gallery.**

- Initially you need to have the permission

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

- Use the Following Code to have the layout as designed follows.





```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:paddingBottom="@dimen/activity_vertical_margin"
 android:paddingLeft="@dimen/activity_horizontal_margin"
 android:paddingRight="@dimen/activity_horizontal_margin"
 android:paddingTop="@dimen/activity_vertical_margin"
 tools:context="androidexamples.idevroids.loadimagefrmgallery.MainActivity">

 <ImageView
 android:id="@+id/imgView"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:background="@color/abc_search_url_text_normal"></ImageView>

 <Button
 android:id="@+id/buttonLoadPicture"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="0"
 android:text="Load Picture"
 android:layout_gravity="bottom|center"></Button>

</LinearLayout>

```

3. 使用以下代码通过按钮点击显示图片。

按钮点击将会是

```

Button loadImg = (Button) this.findViewById(R.id.buttonLoadPicture);
loadImg.setOnClickListener(new View.OnClickListener() {
 @Override
 public void 点击事件(View v) {

Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
startActivityForResult(i, RESULT_LOAD_IMAGE);
}
});

```

3. 一旦你点击按钮，它将通过意图打开图库。

你需要选择图片并将其发送回主活动。这里通过 `onActivityResult` 可以实现。

```

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);

 if (requestCode == RESULT_LOAD_IMAGE && resultCode == RESULT_OK && null != data) {
 Uri selectedImage = data.getData();
 String[] filePathColumn = { MediaStore.Images.Media.DATA };

 Cursor cursor = getContentResolver().query(selectedImage,
 filePathColumn, null, null, null);
 cursor.moveToFirst();

 int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
 String picturePath = cursor.getString(columnIndex);
 cursor.close();
 }
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:paddingBottom="@dimen/activity_vertical_margin"
 android:paddingLeft="@dimen/activity_horizontal_margin"
 android:paddingRight="@dimen/activity_horizontal_margin"
 android:paddingTop="@dimen/activity_vertical_margin"
 tools:context="androidexamples.idevroids.loadimagefrmgallery.MainActivity">

 <ImageView
 android:id="@+id/imgView"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:background="@color/abc_search_url_text_normal"></ImageView>

 <Button
 android:id="@+id/buttonLoadPicture"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="0"
 android:text="Load Picture"
 android:layout_gravity="bottom|center"></Button>

</LinearLayout>

```

3. Use the Following code to Display the image with button Click.

Button Click will be

```

Button loadImg = (Button) this.findViewById(R.id.buttonLoadPicture);
loadImg.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {

 Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
 startActivityForResult(i, RESULT_LOAD_IMAGE);
 }
});

```

3. Once you clicked on the button , it will open the gallery with help of intent.

You need to select image and send it back to main activity. Here with help of `onActivityResult` we can do that.

```

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);

 if (requestCode == RESULT_LOAD_IMAGE && resultCode == RESULT_OK && null != data) {
 Uri selectedImage = data.getData();
 String[] filePathColumn = { MediaStore.Images.Media.DATA };

 Cursor cursor = getContentResolver().query(selectedImage,
 filePathColumn, null, null, null);
 cursor.moveToFirst();

 int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
 String picturePath = cursor.getString(columnIndex);
 cursor.close();
 }
}

```

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
 imageView.setImageBitmap(BitmapFactory.decodeFile(picturePath));
}
}
```

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
 imageView.setImageBitmap(BitmapFactory.decodeFile(picturePath));
}
}
```

# 第262章：异常

## 第262.1节：ActivityNotFoundException

这是一种非常常见的异常。它会导致您的应用在启动或执行过程中停止。在LogCat中您会看到如下信息：

```
android.content.ActivityNotFoundException: 无法找到明确的活动类；
您是否在AndroidManifest.xml中声明了该活动？
```

在这种情况下，请检查您是否在AndroidManifest.xml文件中声明了您的活动。

在AndroidManifest.xml中声明您的Activity的最简单方法是：

```
<activity android:name="com.yourdomain.YourStoppedActivity" />
```

## 第262.2节：OutOfMemoryError

这是一个运行时错误，当你在堆上请求大量内存时会发生。这在将位图加载到ImageView中时很常见。

你有以下几个选项：

1. 使用大型应用堆

在你的 AndroidManifest.xml 文件中的 application 标签中添加 "largeHeap" 选项。这将为你的应用提供更多内存，但可能无法解决根本问题。

```
<application largeHeap="true" ... >
```

2. 回收你的位图

加载位图后，确保回收它以释放内存：

```
if (bitmap != null && !bitmap.isRecycled())
 bitmap.recycle();
```

3. 加载采样位图到内存

避免一次性将整个位图加载到内存中，通过使用 BitmapOptions 和

inSampleSize 采样较小的尺寸。

参见 [Android 文档](#) 了解示例

## 第262.3节：为未预料的异常注册自定义处理器

这是您如何对未被捕获的异常做出反应，类似于系统的标准“应用程序 XYZ 已崩溃”提示

```
import android.app.Application;
import android.util.Log;

import java.io.File;
```

# Chapter 262: Exceptions

## Section 262.1: ActivityNotFoundException

This is a very common [Exception](#). It causes your application to stop during the start or execution of your app. In the LogCat you see the message:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

In this case, check if you have declared your activity in the [AndroidManifest.xml](#) file.

The simplest way to declare your Activity in [AndroidManifest.xml](#) is:

```
<activity android:name="com.yourdomain.YourStoppedActivity" />
```

## Section 262.2: OutOfMemoryError

This is a runtime error that happens when you request a large amount of memory on the heap. This is common when loading a Bitmap into an ImageView.

You have some options:

1. Use a large application heap

Add the "largeHeap" option to the application tag in your [AndroidManifest.xml](#). This will make more memory available to your app but will likely not fix the root issue.

```
<application largeHeap="true" ... >
```

2. Recycle your bitmaps

After loading a bitmap, be sure to recycle it and free up memory:

```
if (bitmap != null && !bitmap.isRecycled())
 bitmap.recycle();
```

3. Load sampled bitmaps into memory

Avoid loading the entire bitmap into memory at once by sampling a reduced size, using [BitmapOptions](#) and [inSampleSize](#).

See [Android documentation](#) for example

## Section 262.3: Registering own Handler for unexpected exceptions

This is how you can react to exceptions which have not been caught, similar to the system's standard “Application XYZ has crashed”

```
import android.app.Application;
import android.util.Log;

import java.io.File;
```

```

import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * 应用程序类，在崩溃前将未预料的异常写入崩溃文件。
 */
public class MyApplication extends Application {
 private static final String TAG = "ExceptionHandler";

 @Override
 public void onCreate() {
 super.onCreate();

 // 设置未捕获异常的处理器。
 final Thread.UncaughtExceptionHandler defaultHandler =
 Thread.getDefaultUncaughtExceptionHandler();
 Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
 @Override
 public void uncaughtException(Thread thread, Throwable e) {
 try {
 handleUncaughtException(e);
 System.exit(1);
 } catch (Throwable e2) {
 Log.e(TAG, "Exception in custom exception handler", e2);
 defaultHandler.uncaughtException(thread, e);
 }
 }
 });
 }

 private void handleUncaughtException(Throwable e) throws IOException {
 Log.e(TAG, "Uncaught exception logged to local file", e);

 // 创建一个新的唯一文件
 final DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss", Locale.US);
 String timestamp;
 File file = null;
 while (file == null || file.exists()) {
 timestamp = dateFormat.format(new Date());
 file = new File(getFilesDir(), "crashLog_" + timestamp + ".txt");
 }
 Log.i(TAG, "尝试创建日志文件 " + file.getPath());
 file.createNewFile();

 // 将堆栈跟踪写入文件
 FileWriter writer = null;
 try {
writer = new FileWriter(file, true);
 for (StackTraceElement element : e.getStackTrace()) {
 writer.write(element.toString());
 }
 } finally {
 if (writer != null) writer.close();
 }

 // 你也可以（并且应该）显示一个对话框来通知用户
 }
}

```

```

import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * Application class writing unexpected exceptions to a crash file before crashing.
 */
public class MyApplication extends Application {
 private static final String TAG = "ExceptionHandler";

 @Override
 public void onCreate() {
 super.onCreate();

 // Setup handler for uncaught exceptions.
 final Thread.UncaughtExceptionHandler defaultHandler =
 Thread.getDefaultUncaughtExceptionHandler();
 Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
 @Override
 public void uncaughtException(Thread thread, Throwable e) {
 try {
 handleUncaughtException(e);
 System.exit(1);
 } catch (Throwable e2) {
 Log.e(TAG, "Exception in custom exception handler", e2);
 defaultHandler.uncaughtException(thread, e);
 }
 }
 });
 }

 private void handleUncaughtException(Throwable e) throws IOException {
 Log.e(TAG, "Uncaught exception logged to local file", e);

 // Create a new unique file
 final DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss", Locale.US);
 String timestamp;
 File file = null;
 while (file == null || file.exists()) {
 timestamp = dateFormat.format(new Date());
 file = new File(getFilesDir(), "crashLog_" + timestamp + ".txt");
 }
 Log.i(TAG, "Trying to create log file " + file.getPath());
 file.createNewFile();

 // Write the stacktrace to the file
 FileWriter writer = null;
 try {
writer = new FileWriter(file, true);
 for (StackTraceElement element : e.getStackTrace()) {
 writer.write(element.toString());
 }
 } finally {
 if (writer != null) writer.close();
 }

 // You can (and probably should) also display a dialog to notify the user
 }
}

```

然后在你的 AndroidManifest.xml 中注册此 Application 类：

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

## 第 262.4 节：未捕获异常 (UncaughtException)

如果您想处理未捕获的异常，尝试在您的应用程序类的 onCreate 方法中捕获所有异常：

```
public class MyApp extends Application {
 @Override
 public void onCreate() {
 super.onCreate();
 try {
 Thread
.setDefaultUncaughtExceptionHandler(
 new Thread.UncaughtExceptionHandler() {

 @Override
 public void uncaughtException(Thread thread, Throwable e) {
 Log.e(TAG,
 "Uncaught Exception thread: "+thread.getName()+"
 "+e.getStackTrace());
 handleUncaughtException (thread, e);
 }
 }
} catch (SecurityException e) {
 Log.e(TAG,
 "Could not set the Default Uncaught Exception Handler:
 "+e.getStackTrace());
}

 private void 处理未捕获异常 (线程 线程, Throwable 异常){
 Log.e(标签, "未捕获异常");
 e.printStackTrace();
 }
}
```

## 第262.5节：NetworkOnMainThreadException (主线程网络异常)

来自 [文档](#)：

当应用尝试在其

主线程上执行网络操作时抛出的异常。

此异常仅针对目标为Honeycomb SDK或更高版本的应用抛出。针对早期 SDK版本的应用允许在其主事件循环线程上进行网络操作，但强烈不建议这样做。

以下是可能导致该异常的代码片段示例：

```
public class 主活动 extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

Then register this Application class in your AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

## Section 262.4: UncaughtException

If you want to handle uncaught exceptions try to catch them all in onCreate method of you Application class:

```
public class MyApp extends Application {
 @Override
 public void onCreate() {
 super.onCreate();
 try {
 Thread
.setDefaultUncaughtExceptionHandler(
 new Thread.UncaughtExceptionHandler() {

 @Override
 public void uncaughtException(Thread thread, Throwable e) {
 Log.e(TAG,
 "Uncaught Exception thread: "+thread.getName()+"
 "+e.getStackTrace());
 handleUncaughtException (thread, e);
 }
 }
} catch (SecurityException e) {
 Log.e(TAG,
 "Could not set the Default Uncaught Exception Handler:
 "+e.getStackTrace());
}

 private void handleUncaughtException (Thread thread, Throwable e){
 Log.e(TAG, "uncaughtException");
 e.printStackTrace();
 }
}
```

## Section 262.5: NetworkOnMainThreadException

From [the documentation](#):

The exception that is thrown when an application attempts to perform a networking operation on its main thread.

This is only thrown for applications targeting the Honeycomb SDK or higher. Applications targeting earlier SDK versions are allowed to do networking on their main event loop threads, but it's heavily discouraged.

Here's an example of a code fragment that may cause that exception:

```
public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

```
Uri.Builder 构建器 = new Uri.Builder().scheme("http").authority("www.google.com");
HttpURLConnection 连接 = null;
BufferedReader 读取器 = null;
URL 网址;
try {
url = new URL(builder.build().toString());
urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.connect();
} catch (IOException e) {
Log.e("TAG", "连接错误", e);
} finally{
 if (urlConnection != null) {
 urlConnection.disconnect();
 }
 if (reader != null) {
 try {
reader.close();
 } catch (final IOException e) {
 Log.e("TAG", "关闭流时出错", e);
 }
 }
}
}
```

上述代码会抛出NetworkOnMainThreadException异常，针对目标为Honeycomb SDK (Android v3.0) 或更高版本的应用，因为应用试图在主线程上执行网络操作。

为避免此异常，您的网络操作必须始终通过AsyncTask、Thread

IntentService等在后台任务中运行。

```
private class MyAsyncTask extends AsyncTask<String, Integer, Void> {
 @Override
 protected Void doInBackground(String[] params) {
 Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
 HttpURLConnection urlConnection = null;
 BufferedReader reader = null;
 URL url;
 try {
 url = new URL(builder.build().toString());
 urlConnection = (HttpURLConnection) url.openConnection();
 urlConnection.setRequestMethod("GET");
 urlConnection.connect();
 } catch (IOException e) {
 Log.e("TAG", "连接错误", e);
 } finally{
 if (urlConnection != null) {
 urlConnection.disconnect();
 }
 if (reader != null) {
 try {
 reader.close();
 } catch (final IOException e) {
 Log.e("TAG", "关闭流时出错", e);
 }
 }
 }
 }
 return null;
}
```

```
Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
HttpURLConnection urlConnection = null;
BufferedReader reader = null;
URL url;
try {
 url = new URL(builder.build().toString());
 urlConnection = (HttpURLConnection) url.openConnection();
 urlConnection.setRequestMethod("GET");
 urlConnection.connect();
} catch (IOException e) {
 Log.e("TAG", "Connection error", e);
} finally{
 if (urlConnection != null) {
 urlConnection.disconnect();
 }
 if (reader != null) {
 try {
 reader.close();
 } catch (final IOException e) {
 Log.e("TAG", "Error closing stream", e);
 }
 }
}
```

Above code will throw `NetworkOnMainThreadException` for applications targeting Honeycomb SDK (Android v3.0) or higher as the application is trying to perform a network operation on the main thread.

To avoid this exception, your network operations must always run in a background task via an `AsyncTask`, `Thread`, `IntentService`, etc.

```
private class MyAsyncTask extends AsyncTask<String, Integer, Void> {

 @Override
 protected Void doInBackground(String[] params) {
 Uri.Builder builder = new Uri.Builder().scheme("http").author
 HttpURLConnection urlConnection = null;
 BufferedReader reader = null;
 URL url;
 try {
 url = new URL(builder.build().toString());
 urlConnection = (HttpURLConnection) url.openConnection();
 urlConnection.setRequestMethod("GET");
 urlConnection.connect();
 } catch (IOException e) {
 Log.e("TAG", "Connection error", e);
 } finally{
 if (urlConnection != null) {
 urlConnection.disconnect();
 }
 if (reader != null) {
 try {
 reader.close();
 } catch (final IOException e) {
 Log.e("TAG", "Error closing stream", e);
 }
 }
 }
 return null;
 }
}
```

```
}
```

## 第262.6节：DexException

com.android.dex.DexException: 多个dex文件定义了Lcom/example/lib/Class;

此错误发生是因为应用在打包时发现了两个.dex文件定义了相同的方法集。

通常这是因为应用意外地对同一个库有了两个独立的依赖。

例如，假设你有一个项目，想依赖两个库：A和B，它们各自有自己的依赖。如果库B已经依赖了库A，当库A单独被添加到项目中时，就会抛出此错误。编译库B时已经包含了来自A的代码集，所以当编译器去打包库A时，发现库A的方法已经被打包了。

为了解决此问题，请确保你的依赖中没有任何可能被意外重复添加的情况。

```
}
```

## Section 262.6: DexException

com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;

This error occurs because the app, when packaging, finds two .dex files that define the same set of methods.

Usually this happens because the app has accidentally acquired 2 separate dependencies on the same library.

For instance, say you have a project, and you want to rely on two libraries: A and B, which each have their own dependencies. If library B already has a dependency on library A, this error will be thrown if library A is added to the project by itself. Compiling library B already gave the set of code from A, so when the compiler goes to bundle library A, it finds library A's methods already packaged.

To resolve, make sure that none of your dependencies could accidentally be added twice in such a manner

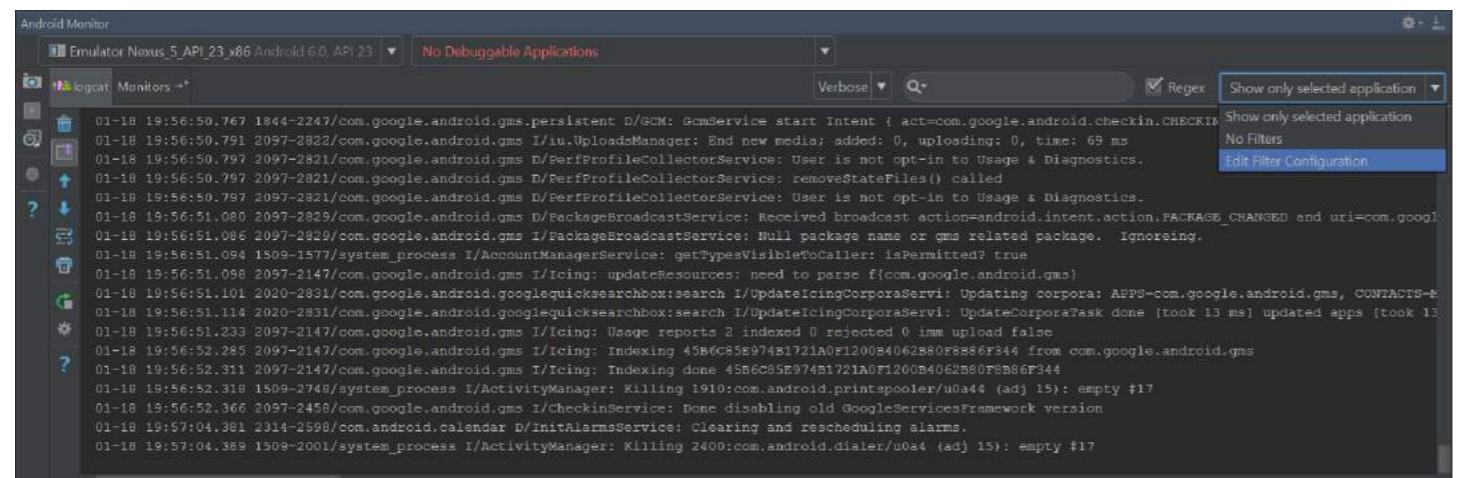
# 第263章：日志记录和使用Logcat

## 选项

选项	描述
-b (缓冲区)	加载备用日志缓冲区以供查看，例如事件或无线电。默认使用主缓冲区。 参见查看替代日志缓冲区。
-c	清除（刷新）整个日志并退出。
-d	将日志转储到屏幕并退出。
-f (filename)	将日志消息输出写入 (filename)。默认是 stdout。
-g	打印指定日志缓冲区的大小并退出。
-n (count)	设置最大轮换日志数量为 (count)。默认值为4。需要 -r 选项。
-r (千字节)	每输出 (千字节) 时轮换日志文件。默认值为16。需要使用-f选项。
-s	将默认过滤器规格设置为静默模式。
-v (格式)	设置日志消息的输出格式。默认格式为简洁格式。

## 第263.1节：过滤logcat输出

过滤logcat输出很有帮助，因为有许多消息并不感兴趣。要过滤输出，打开“Android监视器”，点击右上角的下拉菜单，选择编辑过滤器配置



现在你可以添加自定义过滤器以显示感兴趣的消息，同时过滤掉可以安全忽略的已知日志行。要忽略输出的一部分，可以定义正则表达式。以下是排除匹配标签的示例：

```
^(?!(HideMe|AndThis))
```

可以按照以下示例输入：

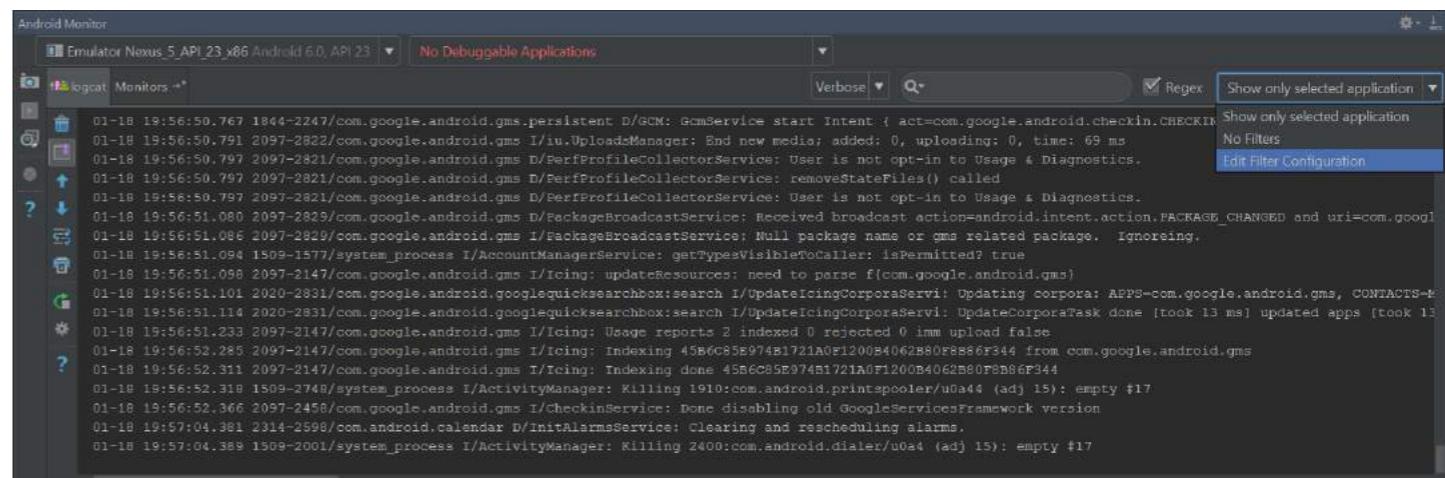
# Chapter 263: Logging and using Logcat

## Option

Option	Description
-b (buffer)	Loads an alternate log buffer for viewing, such as events or radio. The main buffer is used by default. See Viewing Alternative Log Buffers.
-c	Clears (flushes) the entire log and exits.
-d	Dumps the log to the screen and exits.
-f (filename)	Writes log message output to (filename). The default is stdout.
-g	Prints the size of the specified log buffer and exits.
-n (count)	Sets the maximum number of rotated logs to (count). The default value is 4. Requires the -r option.
-r (kbytes)	Rotates the log file every (kbytes) of output. The default value is 16. Requires the -f option.
-s	Sets the default filter spec to silent.
-v (format)	Sets the output format for log messages. The default is brief format.

## Section 263.1: Filtering the logcat output

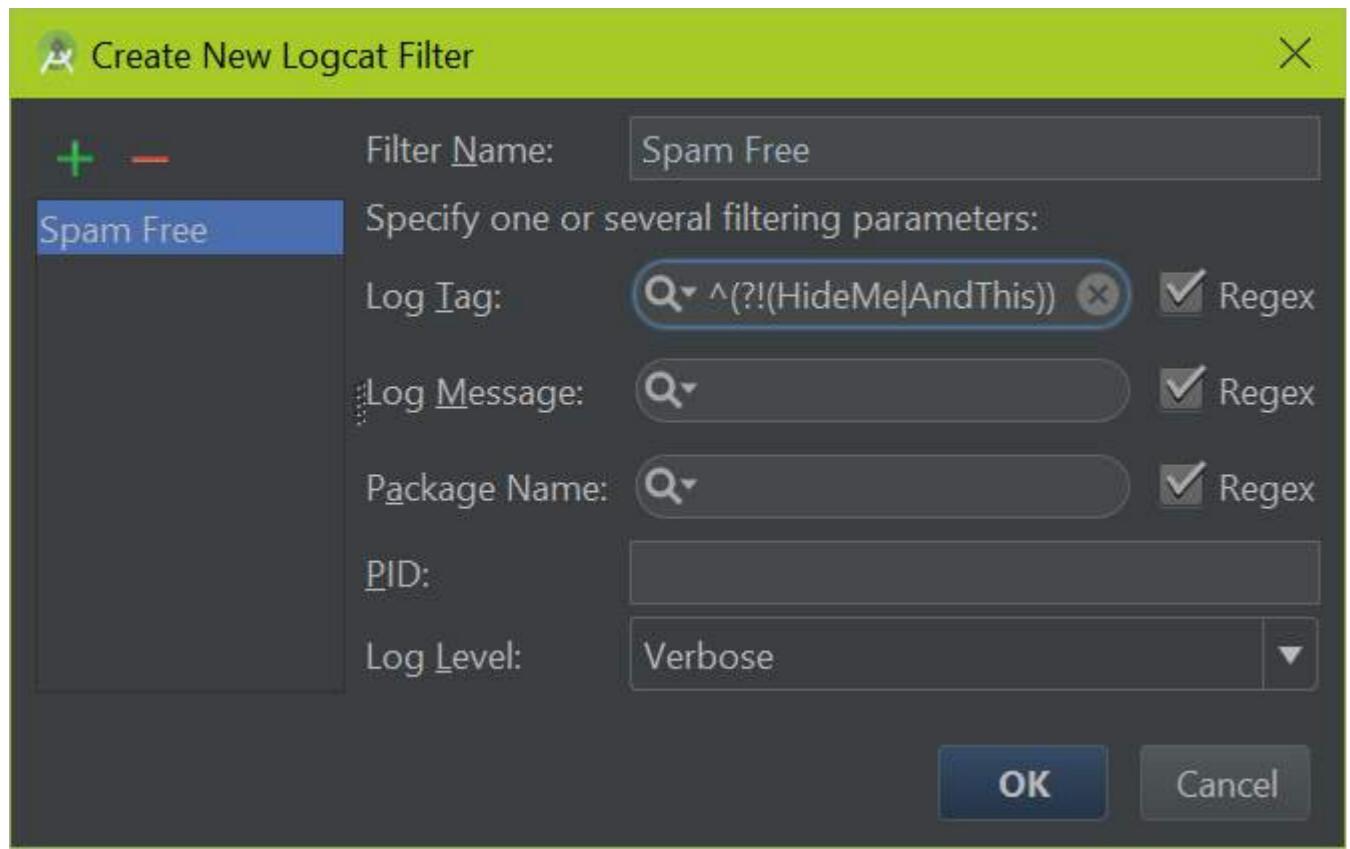
It is helpful to filter the logcat output because there are many messages which are not of interest. To filter the output, open the "Android Monitor" and click on the drop down on the top-right and select *Edit Filter Configuration*



Now you can add custom filters to show messages which are of interest, as well as filter out well-known log lines which can safely be ignored. To ignore a part of the output you may define a Regular Expression. Here is an example of excluding matching tags:

```
^(?!(HideMe|AndThis))
```

This can be entered by following this example:



以上是一个排除输入的正则表达式。如果你想向黑名单中添加另一个标签，可以在管道符号 | 后添加。例如，如果你想将“GC”加入黑名单，可以使用如下过滤器：

```
^(?!(HideMe|AndThis|GC))
```

更多文档和示例请访问 [Logging and using Logcat](#)

## 第263.2节：日志记录

任何高质量的安卓应用都会通过应用日志来跟踪其运行情况。这些日志为开发者提供了便捷的调试帮助，以诊断应用的运行状况。完整的安卓文档可以在这里找到，以下是摘要：

### 基础日志记录

Log类是编写开发者日志的主要来源，通过指定标签和消息。标签用于过滤日志消息，以识别哪些行来自你的特定Activity。只需调用

```
Log.v(String tag, String msg);
```

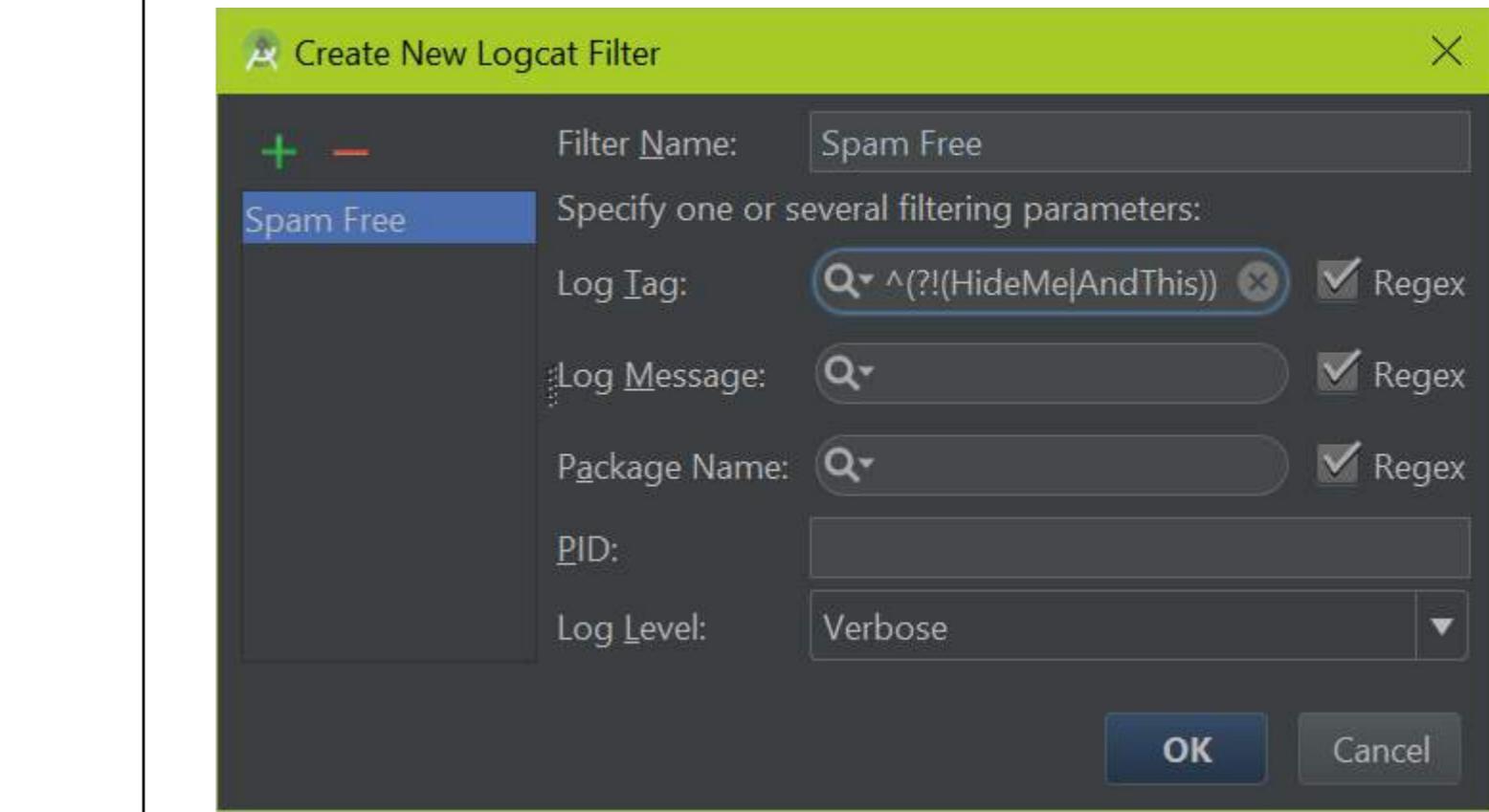
安卓系统就会将消息写入logcat：

```
07-28 12:00:00.759 24812-24839/my.packagename V/MyAnimator: Some log messages
└ time stamp | app.package | any tag |
 process & thread ids | log level | the log message
```

提示：

注意进程ID和线程ID。如果它们相同时——日志来自主线程/UI线程！

可以使用任意标签，但通常使用类名作为标签：



The above is a regular expression which excludes inputs. If you wanted to add another tag to the *blacklist*, add it after a pipe | character. For example, if you wanted to blacklist "GC", you would use a filter like this:

```
^(?!(HideMe|AndThis|GC))
```

For more documentation and examples visit [Logging and using Logcat](#)

## Section 263.2: Logging

Any quality Android application will keep track of what it's doing through application logs. These logs allow easy debugging help for the developer to diagnose what's going on with the application. Full Android Documentation can be found [here](#), but a summary follows:

### Basic Logging

The Log class is the main source of writing developer logs, by specifying a tag and a message. The tag is what you can use to filter log messages by to identify which lines come from your particular Activity. Simply call

```
Log.v(String tag, String msg);
```

And the Android system will write a message to the logcat:

```
07-28 12:00:00.759 24812-24839/my.packagename V/MyAnimator: Some log messages
└ time stamp | app.package | any tag |
 process & thread ids | log level | the log message
```

TIP:

Notice the process id and the thread id. If they are the same - the log is coming from the main/UI thread!

Any tag can be used, but it is common to use the class name as a tag:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

## 日志级别

Android日志记录器有6个不同的级别，每个级别都有特定的用途：

- ERROR: Log.e()
  - 用于表示严重错误，当抛出Exception时会打印此级别。
- 警告: Log.w()
  - 用于指示警告，主要针对可恢复的故障
- INFO: Log.i()
  - 用于指示关于应用程序状态的更高级别信息
- DEBUG: Log.d()
  - 用于记录在调试应用程序时有用的信息，但在运行应用程序时会造成干扰
- VERBOSE: Log.v()
  - 用于记录反映应用程序状态细节的信息
- ASSERT: Log.wtf()
  - 用于记录不应该发生的情况的信息。
  - wtf代表“多么糟糕的失败”(What a Terrible Failure)。

## 记录的动机

记录的动机是通过查看应用程序事件链，轻松找到错误、警告和其他信息。例如，假设有一个应用程序从文本文件读取行，但错误地假设文件永远不会为空。未记录日志的应用程序的日志跟踪可能如下所示：

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
 com.example.SomeRandomException: 预期字符串，实际得到'null'
```

接着是一堆堆栈跟踪，最终会指向出错的代码行，通过调试器逐步跟踪最终会定位到问题所在

然而，启用日志记录的应用程序的日志跟踪可能看起来像这样：

```
V/MyApplication: 正在寻找 SD 卡上的文件 myFile.txt
D/MyApplication: 在路径 <path> 找到文件 myFile.txt
V/MyApplication: 正在打开文件 myFile.txt
D/MyApplication: 读取文件 myFile.txt 完成，发现 0 行
V/MyApplication: 正在关闭文件 myFile.txt
...
E/MyApplication: Process: com.example.myapplication, PID: 25788
 com.example.SomeRandomException: 预期字符串，实际得到'null'
```

快速浏览日志，很明显该文件是空的。

## 日志记录时需要考虑的事项：

虽然日志记录是一个强大的工具，允许 Android 开发者更深入地了解其应用程序的内部工作原理，但日志记录也存在一些缺点。

## 日志可读性：

Android 应用程序通常会同时运行多个日志。因此，确保每个日志易于阅读且仅包含相关且必要的信息非常重要。

## 性能：

日志记录确实需要少量的系统资源。一般来说，这不值得担心，然而，如果

```
public static final String tag = MyAnimator.class.getSimpleName();
```

## Log Levels

The Android logger has 6 different levels, each of which serve a certain purpose:

- ERROR: Log.e()
  - Used to indicate critical failure, this is the level printed at when throwing an Exception.
- WARN: Log.w()
  - Used to indicate a warning, mainly for recoverable failures
- INFO: Log.i()
  - Used to indicate higher-level information about the state of the application
- DEBUG: Log.d()
  - Used to log information that would be useful to know when debugging the application, but would get in the way when running the application
- VERBOSE: Log.v()
  - Used to log information that reflects the small details about the state of the application
- ASSERT: Log.wtf()
  - Used to log information about a condition that should never happen.
  - wtf stands for "What a Terrible Failure".

## Motivation For Logging

The motivation for logging is to easily find errors, warnings, and other information by glancing at the chain of events from the application. For instance, imagine an application that reads lines from a text file, but incorrectly assumes that the file will never be empty. The log trace (of an app that doesn't log) would look something like this:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
 com.example.SomeRandomException: Expected string, got 'null' instead
```

Followed by a bunch of stack traces that would eventually lead to the offending line, where stepping through with a debugger would eventually lead to the problem

However, the log trace of an application with logging enabled could look something like this:

```
V/MyApplication: Looking for file myFile.txt on the SD card
D/MyApplication: Found file myFile.txt at path <path>
V/MyApplication: Opening file myFile.txt
D/MyApplication: Finished reading myFile.txt, found 0 lines
V/MyApplication: Closing file myFile.txt
...
E/MyApplication: Process: com.example.myapplication, PID: 25788
 com.example.SomeRandomException: Expected string, got 'null' instead
```

A quick glance at the logs and it is obvious that the file was empty.

## Things To Considering When Logging:

Although logging is a powerful tool that allows Android developers to gain a greater insight into the inner working of their application, logging does have some drawbacks.

## Log Readability:

It is common for Android Applications to have several logs running synchronously. As such, it is very important that each log is easily readable and only contains relevant, necessary information.

## Performance:

Logging does require a small amount of system resources. In general, this does not warrant concern, however, if

过度使用日志记录可能会对应用性能产生负面影响。

#### 安全性：

最近，谷歌应用商店中添加了几款安卓应用，允许用户查看所有正在运行应用的日志。这种无意的数据展示可能使用户查看机密信息。

作为经验法则，发布应用到市场之前，务必删除包含非公开数据的日志*before*发布您的应用到市场。

#### 结论：

日志记录是安卓应用的重要组成部分，因为它赋予开发者强大的能力。创建有用的日志跟踪是软件开发中最具挑战性的方面之一，但安卓的Log类使这变得更加容易。

更多文档和示例请访问 Logging and using Logcat

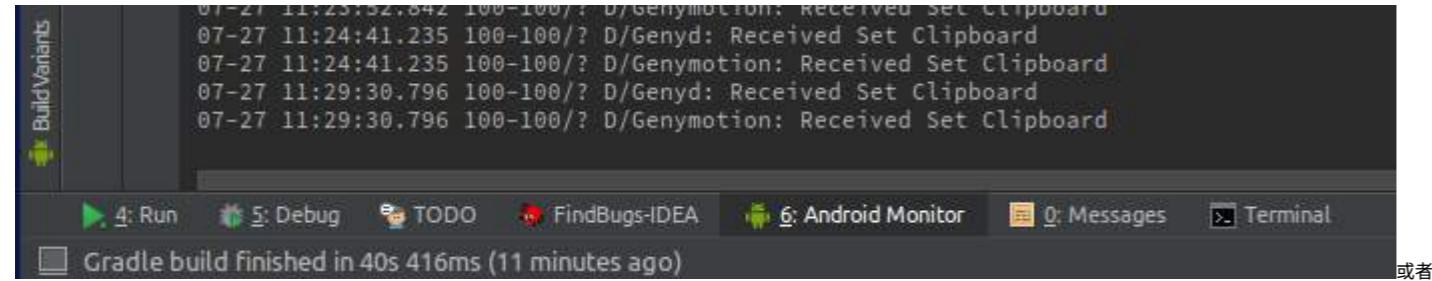
## 第263.3节：使用Logcat

Logcat是一个命令行工具，用于导出系统消息日志，包括设备抛出错误时的堆栈跟踪以及您通过Log类从应用中写入的消息。

Logcat输出可以在Android Studio的Android Monitor中显示，也可以通过adb命令行查看。

#### 在 Android Studio 中

通过点击“Android 监视器”图标显示：



或者

通过按下 **Alt** + **6** 在 Windows/Linux 上，或 **CMD** + **6** 在 Mac 上。

#### 通过命令行：

简单用法：

```
$ adb logcat
```

带时间戳：

```
$ adb logcat -v time
```

按特定文本过滤：

```
$ adb logcat -v time | grep 'searchtext'
```

命令行 logcat 有许多可用的选项和过滤器，文档见 [here](#)。

一个简单但实用的示例是以下过滤表达式，它显示所有标签中优先级为“错误”的日志消息：

```
$ adb logcat *:E
```

overused, logging may have a negative impact on application performance.

#### Security:

Recently, several Android Applications have been added to the Google Play marketplace that allow the user to view logs of all running applications. This unintended display of data may allow users to view confidential information. As a rule of thumb, always remove logs that contain non-public data *before* publishing your application to the marketplace.

#### Conclusion:

Logging is an essential part of an Android application, because of the power it gives to developers. The ability to create a useful log trace is one of the most challenging aspects of software development, but Android's Log class helps to make it much easier.

For more documentation and examples visit Logging and using Logcat

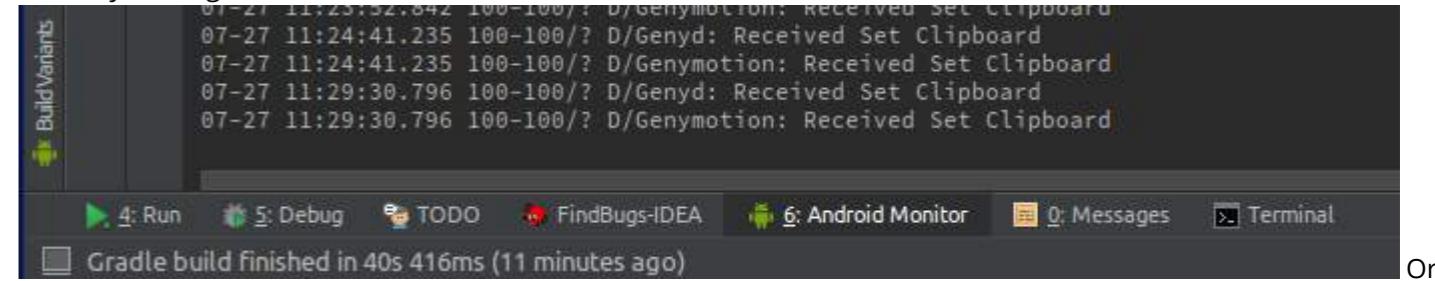
## Section 263.3: Using the Logcat

Logcat is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the Log class.

The Logcat output can be displayed within Android Studio's Android Monitor or with adb command line.

#### In Android Studio

Show by clicking the "Android Monitor" icon:



Or

by pressing **Alt** + **6** on Windows/Linux or **CMD** + **6** on Mac.

#### via command line:

Simple usage:

```
$ adb logcat
```

With timestamps:

```
$ adb logcat -v time
```

Filter on specific text:

```
$ adb logcat -v time | grep 'searchtext'
```

There are many options and filters available to *command line logcat*, documented [here](#).

A simple but useful example is the following filter expression that displays all log messages with priority level "error", on all tags:

```
$ adb logcat *:E
```

## 第263.4节：直接从 Logcat 链接到源代码的日志

这是一个很好的技巧，可以添加代码链接，方便跳转到发出日志的代码。

使用以下代码，这个调用：

```
MyLogger.logWithLink("MyTag","param="+param);
```

将产生：

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3) (MyFrag.java:2366) // << logcat 将其转换为指向源码的链接!
```

这是代码（在一个名为 MyLogger 的类中）：

```
static StringBuilder sb0 = new StringBuilder(); // 可重用的字符串对象

public static void logWithLink(String TAG, Object param) {
 StackTraceElement stack = Thread.currentThread().getStackTrace()[3];
 sb0.setLength(0);
 String c = stack.getFileName().substring(0, stack.getFileName().length() - 5); // 去掉
 ".java"
 sb0.append(c).append(":");
 sb0.append(stack.getMethodName()).append(' ');
 if (param != null) {
 sb0.append(param);
 }
 sb0.append(" ");
 sb0.append(
 "().append(stack.getFileName()).append(':').append(stack.getLineNumber()).append('')");
 Log.d(TAG, sb0.toString());
}
```

这是一个基本示例，可以轻松扩展以发出指向调用者的链接（提示：堆栈将是[4]而不是[3]），  
你也可以添加其他相关信息。

## 第263.5节：清除日志

为了清除（刷新）整个日志：

```
adb logcat -c
```

## 第263.6节：Android Studio的使用

1. 隐藏/显示打印信息：

## Section 263.4: Log with link to source directly from Logcat

This is a nice trick to add a link to code, so it will be easy to jump to the code that issued the log.

With the following code, this call:

```
MyLogger.logWithLink("MyTag", "param="+param);
```

Will result in:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3) (MyFrag.java:2366) // << logcat converts
this to a link to source!
```

This is the code (inside a class called MyLogger):

```
static StringBuilder sb0 = new StringBuilder(); // reusable string object

public static void logWithLink(String TAG, Object param) {
 StackTraceElement stack = Thread.currentThread().getStackTrace()[3];
 sb0.setLength(0);
 String c = stack.getFileName().substring(0, stack.getFileName().length() - 5); // removes the
 ".java"
 sb0.append(c).append(":");
 sb0.append(stack.getMethodName()).append(' ');
 if (param != null) {
 sb0.append(param);
 }
 sb0.append(" ");
 sb0.append(
 "().append(stack.getFileName()).append(':').append(stack.getLineNumber()).append('')");
 Log.d(TAG, sb0.toString());
}
```

This is a basic example, it can be easily extended to issue a link to the caller (hint: the stack will be [4] instead of [3]),  
and you can also add other relevant information.

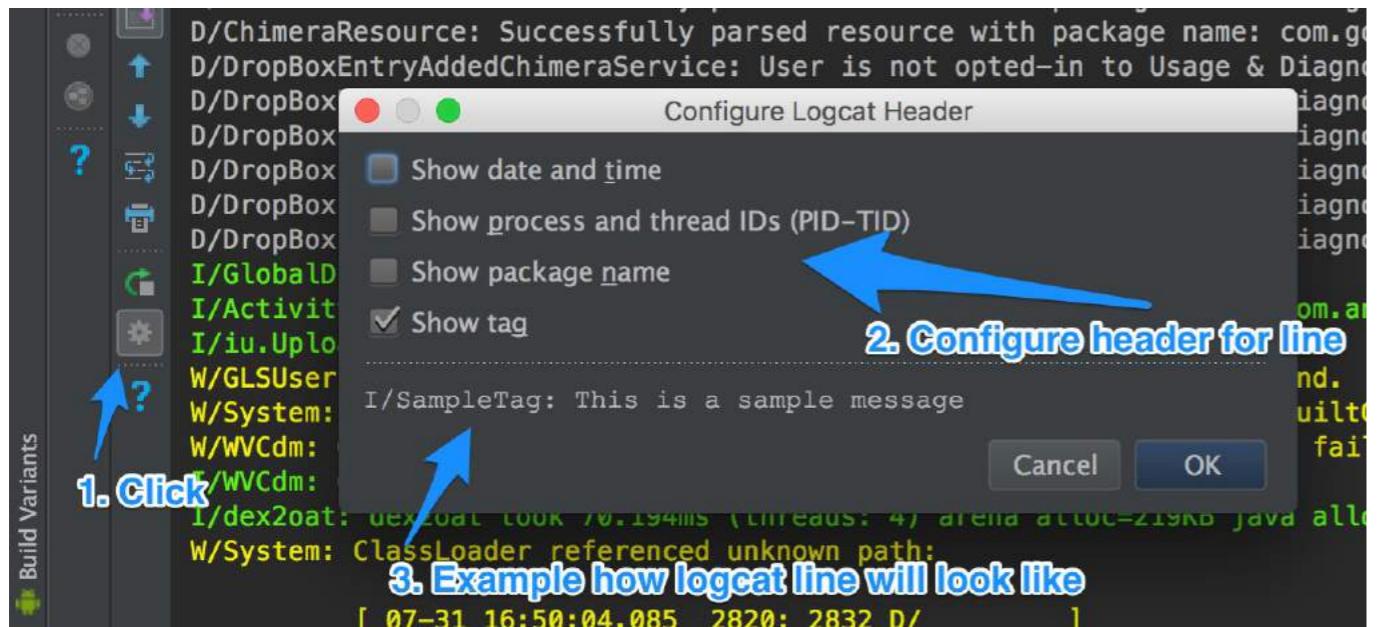
## Section 263.5: Clear logs

In order to clear (flush) the entire log:

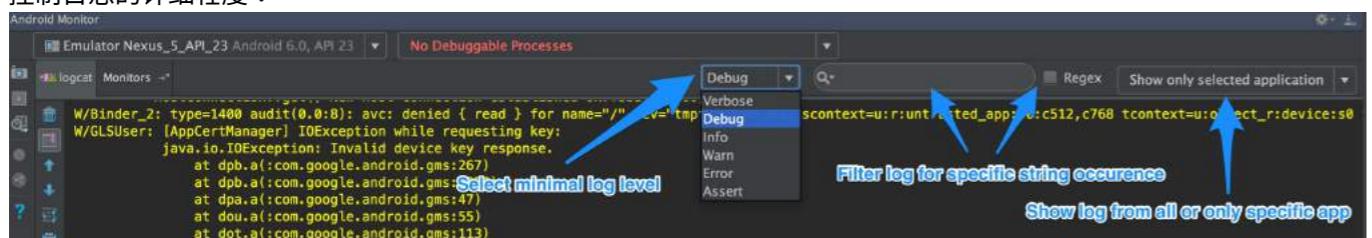
```
adb logcat -c
```

## Section 263.6: Android Studio usage

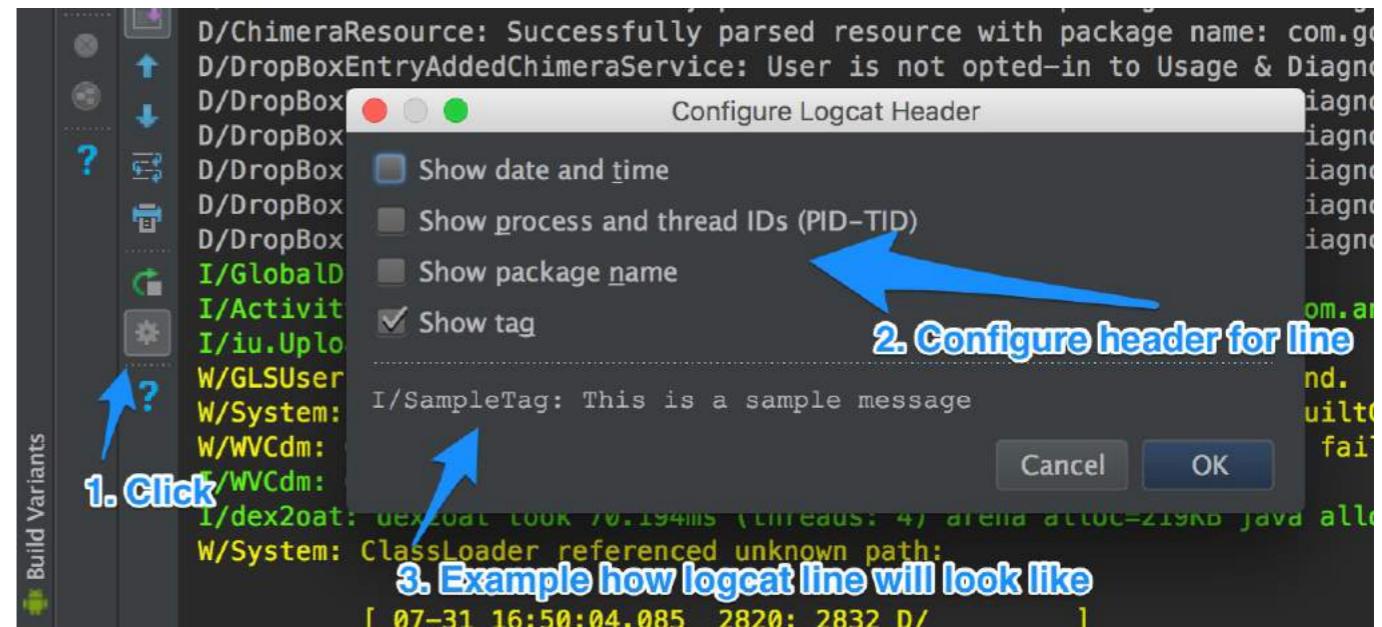
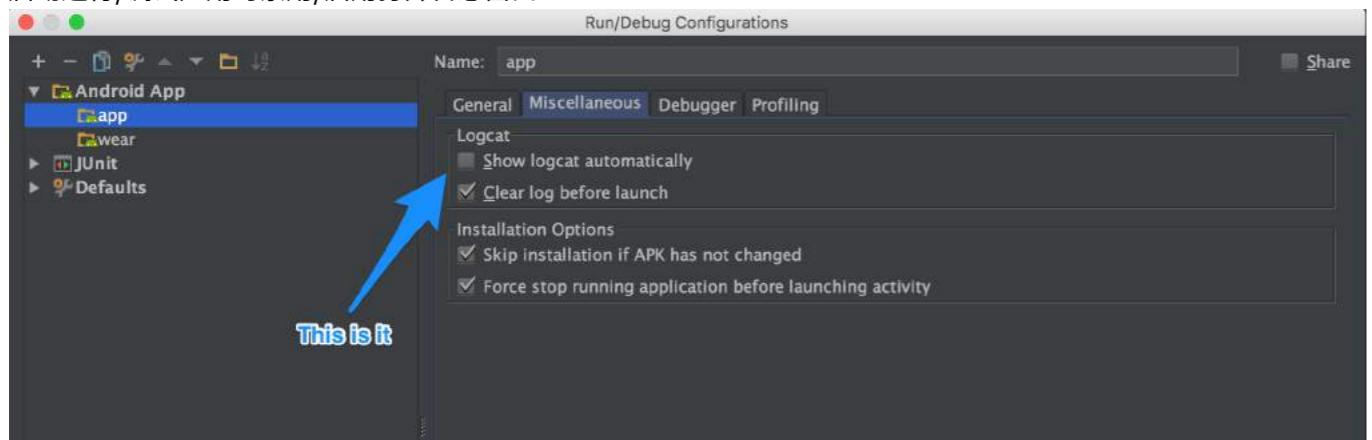
1. Hide/show printed information:



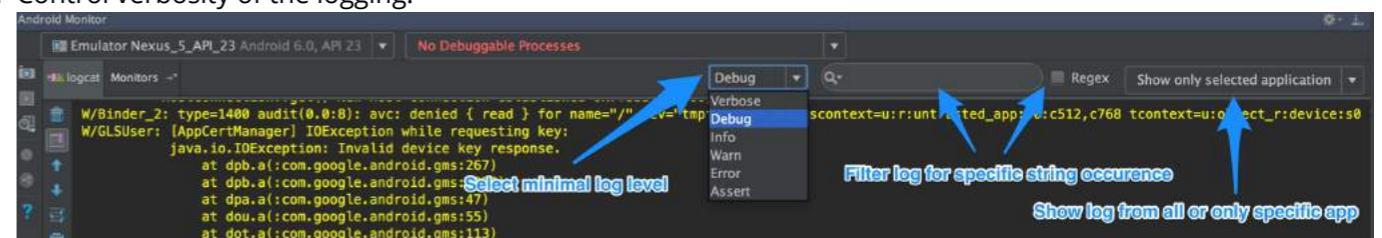
## 2. 控制日志的详细程度：



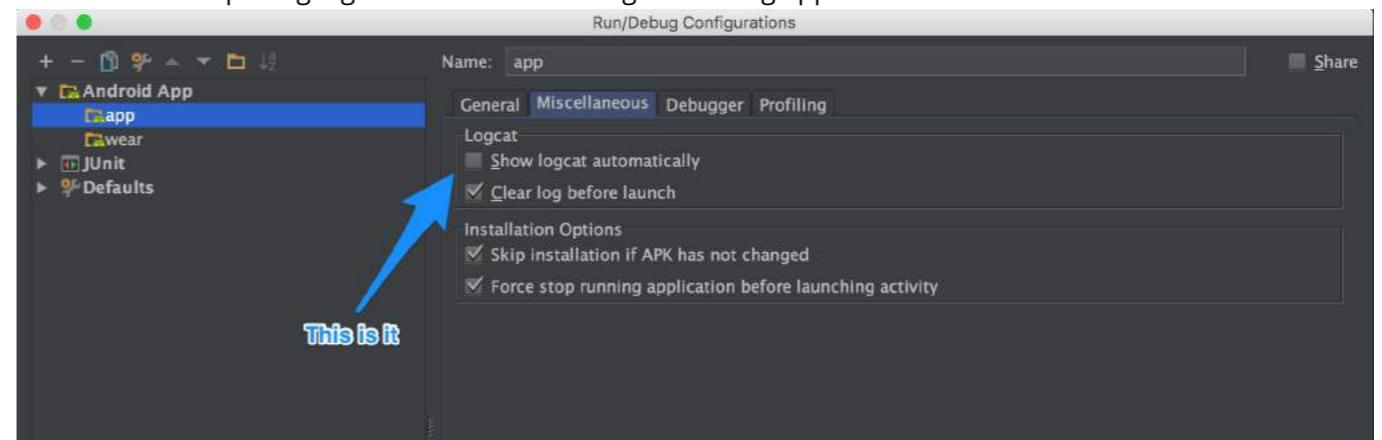
## 3. 启动运行/调试应用时禁用/启用打开日志窗口



## 2. Control verbosity of the logging:



## 3. Disable/enable opening log window when starting run/debug application



## 第263.7节：生成日志代码

Android Studio的Live templates可以提供许多快捷方式来快速记录日志。

使用Live templates，只需开始输入模板名称，然后按TAB或回车键即可插入该语句。

示例：

- logi → 变为 → android.util.Log.i(TAG, "\$METHOD\_NAME\$: \$content\$");\$METHOD\_NA
  - ME\$会自动替换为你的方法名，光标会等待填写内容。
- loge → 同理，用于错误日志
- 其余日志级别亦是如此。

完整的模板列表可在Android Studio的设置中找到 ( + 和类型“live”）。并且可以

## Section 263.7: Generating Logging code

Android Studio's Live templates can offer quite a few shortcuts for quick logging.

To use Live templates, all you need to do is to start typing the template name, and hit TAB or enter to insert the statement.

Examples:

- logi → turns into → android.util.Log.i(TAG, "\$METHOD\_NAME\$: \$content\$");
  - \$METHOD\_NAME\$ will automatically be replaced with your method name, and the cursor will wait for the content to be filled.
- loge → same, for error
- etc. for the rest of the logging levels.

Full list of templates can be found in Android Studio's settings ( + and type "live"). And it is possible to

添加您自定义的模板。

如果您觉得Android Studio的Live templates不能满足您的需求，可以考虑[Android Postfix Plugin](#)

这是一个非常有用的库，帮助您避免手动编写日志行。

语法非常简单：

.log - 日志记录。如果存在常量变量“TAG”，则使用“TAG”。否则使用类名。

```
ublic class MyActivity extends Activity {
 static final String TAG = "test";
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 //
 }
 };
 }
}
```

add your custom templates as well.

If you find Android Studio's Live templates not enough for your needs, you can consider [Android Postfix Plugin](#)

This is a very useful library which helps you to avoid writing the logging line manually.

The syntax is absolutely simple:

.log - Logging. If there is constant variable "TAG", it use "TAG" . Else it use class name.

```
ublic class MyActivity extends Activity {
 static final String TAG = "test";
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 //
 }
 };
 }
}
```

# 第264章：ADB (Android调试桥)

ADB (Android调试桥) 是一个命令行工具，用于与模拟器实例或连接的Android设备通信。

## ADB概述

本主题的大部分内容已拆分至 adb shell

## 第264.1节：通过WiFi连接ADB到设备

标准的ADB配置涉及通过USB连接到物理设备。

如果你愿意，可以切换到TCP/IP模式，通过WiFi连接ADB。

### 未获取root权限的设备

#### 1. 连接到同一网络：

- 确保你的设备和电脑处于同一网络。

#### 2. 使用USB线将设备连接到主机电脑。

#### 3. 通过网络连接adb到设备：

当你的设备通过USB连接到adb时，执行以下命令以监听某个端口（默认5555）上的TCP/IP连接：

- 输入adb tcpip <port> (切换到TCP/IP模式)。
- 断开目标设备的USB线缆。
- 输入 adb connect <ip 地址>:<端口> (端口可选，默认5555)。

例如：

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

如果你不知道设备的IP，可以：

- 在设备的WiFi设置中查看IP地址。
- 使用ADB通过USB发现IP地址：
  - 通过USB将设备连接到电脑
  - 在命令行中输入 adb shell ifconfig 并复制设备的IP地址

要切换回通过USB调试，请使用以下命令：

```
adb usb
```

你也可以通过安装Android Studio的插件来通过WiFi连接ADB。为此，请进入设置 > 插件并浏览仓库，搜索ADB WiFi，安装它，然后重新打开Android Studio。你会在工具栏中看到一个新的图标，如下图所示。通过USB将设备连接到主机电脑，点击这个Android WiFi ADB图标。它会显示设备是否已连接的消息。一旦连接成功，你就可以拔掉USB线。

# Chapter 264: ADB (Android Debug Bridge)

ADB (Android Debug Bridge) is a command line tool that used to communicate with an emulator instance or connected Android device.

## Overview of ADB

A large portion of this topic was split out to adb shell

## Section 264.1: Connect ADB to a device via WiFi

The standard ADB configuration involves a USB connection to a physical device.

If you prefer, you can switch over to TCP/IP mode, and connect ADB via WiFi instead.

### Not rooted device

#### 1. Get on the same network:

- Make sure your device and your computer are on the same network.

#### 2. Connect the device to the host computer with a USB cable.

#### 3. Connect adb to device over network:

While your device is connected to adb via USB, do the following command to listen for a TCP/IP connection on a port (default 5555):

- Type adb tcpip <port> (switch to TCP/IP mode).
- Disconnect the USB cable from the target device.
- Type adb connect <ip address>:<port> (port is optional; default 5555).

For example:

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

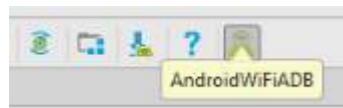
If you don't know your device's IP you can:

- check the IP in the WiFi settings of your device.
- use ADB to discover IP (via USB):
  - Connect the device to the computer via USB
  - In a command line, type adb shell ifconfig and copy your device's IP address

To revert back to debugging via USB use the following command:

```
adb usb
```

You can also connect ADB via WiFi by installing a plugin to Android Studio. In order to do so, go to Settings > Plugins and Browse repositories, search for ADB WiFi, install it, and reopen Android Studio. You will see a new icon in your toolbar as shown in the following image. Connect the device to the host computer via USB and click on this Android WiFi ADB icon. It will display a message whether your device is connected or not. Once it gets connected you can unplug your USB.



#### 已获取Root权限的设备

注意：某些已获取Root权限的设备可以使用Play商店中的ADB WiFi应用程序，以简单的方式启用此功能。此外，对于某些设备（尤其是带有CyanogenMod ROM的设备），此选项存在于设置中的开发者选项中。启用后，会显示连接adb所需的IP地址和端口号，只需执行adb connect <ip地址>:<端口>即可连接。

#### 当你有一台已获取Root权限的设备但没有USB线时

该过程在以下回答中有详细说明：

<http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727>

下面显示了最重要的命令。

在设备上打开终端并输入以下命令：

```
su
setprop service.adb.tcp.port <a tcp port number>
stop adbd
启动 adbd
```

例如：

```
setprop service.adb.tcp.port 5555
```

在你的电脑上：

```
adb connect <ip 地址>:<一个 tcp 端口号>
```

例如：

```
adb connect 192.168.1.2:5555
```

关闭它：

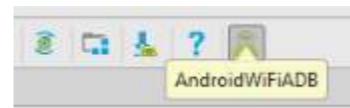
```
setprop service.adb.tcp.port -1
stop adbd
启动 adbd
```

避免超时

默认情况下，adb 会在 5000 毫秒后超时。在某些情况下，如 WiFi 速度慢或 APK 文件较大时，可能会发生这种情况。

在 Gradle 配置中做一个简单的修改即可解决：

```
android {
 adbOptions {
 timeOutInMs 10 * 1000
 }
}
```



#### Rooted device

**Note:** Some devices which **are rooted** can use the ADB WiFi App from the Play Store to enable this in a simple way. Also, for certain devices (especially those with CyanogenMod ROMs) this option is present in the Developer Options among the Settings. Enabling it will give you the IP address and port number required to connect to adb by simply executing adb connect <ip address>:<port>.

#### When you have a rooted device but don't have access to a USB cable

The process is explained in detail in the following answer:

<http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727>

The most important commands are shown below.

Open a terminal in the device and type the following:

```
su
setprop service.adb.tcp.port <a tcp port number>
stop adbd
start adbd
```

For example:

```
setprop service.adb.tcp.port 5555
```

And on your computer:

```
adb connect <ip address>:<a tcp port number>
```

For example:

```
adb connect 192.168.1.2:5555
```

To turn it off:

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

Avoid timeout

By default adb will timeout after 5000 ms. This can happen in some cases such as slow WiFi or large APK.

A simple change in the Gradle configuration can do the trick:

```
android {
 adbOptions {
 timeOutInMs 10 * 1000
 }
}
```

## 第264.2节：在多设备环境中向特定设备发送直接ADB命令

### 1. 通过序列号定位设备

使用-s选项，后跟设备名称，以选择adb命令应在哪个设备上运行。 -s 选项应放在命令之前的第一位。

```
adb -s <device> <command>
```

示例：

```
adb devices

已连接设备列表
emulator-5554 device
02157df2d1faeb33 device

adb -s emulator-5554 shell
```

示例#2：

```
adb devices -l

连接设备列表
06157df65c6b2633 设备 usb:1-3 产品:zeroflte 型号:SM_G920F 设备:zeroflte
LC62TB413962 设备 usb:1-5 产品:a50mpg_dug_htc_emea 型号:HTC_Desire_820G_dual_sim
device:htc_a50mpg_dug

adb -s usb:1-3 shell
```

### 2. 目标设备，当只连接一种设备类型时

你可以使用 -e 定位唯一运行的模拟器

```
adb -e <command>
```

或者你可以使用 -d 定位唯一连接的 USB 设备

```
adb -d <command>
```

## 第264.3节：从设备显示屏截取截图和视频（仅限KitKat）

### 截图：选项1（纯 adb）

shell adb 命令允许我们使用设备内置的 shell 执行命令。screencap shell 命令捕获设备当前可见的内容并保存到指定的图像文件中，例如 /sdcard/screen.png:

```
adb shell screencap /sdcard/screen.png
```

然后你可以使用 pull 命令将文件从设备下载到你电脑的当前目录：

## Section 264.2: Direct ADB command to specific device in a multi-device setting

### 1. Target a device by serial number

Use the -s option followed by a device name to select on which device the adb command should run. The -s options should be first in line, **before** the command.

```
adb -s <device> <command>
```

Example:

```
adb devices

List of devices attached
emulator-5554 device
02157df2d1faeb33 device

adb -s emulator-5554 shell
```

Example#2:

```
adb devices -l

List of devices attached
06157df65c6b2633 device usb:1-3 product:zeroflte model:SM_G920F device:zeroflte
LC62TB413962 device usb:1-5 product:a50mpg_dug_htc_emea model:HTC_Desire_820G_dual_sim
device:htc_a50mpg_dug

adb -s usb:1-3 shell
```

### 2. Target a device, when only one device type is connected

You can target the only running emulator with -e

```
adb -e <command>
```

Or you can target the only connected USB device with -d

```
adb -d <command>
```

## Section 264.3: Taking a screenshot and video (for kitkat only) from a device display

### Screen shot: Option 1 (pure adb)

The shell adb command allows us to execute commands using a device's built-in shell. The screencap shell command captures the content currently visible on a device and saves it into a given image file, e.g. /sdcard/screen.png:

```
adb shell screencap /sdcard/screen.png
```

You can then use the pull command to download the file from the device into the current directory on your computer:

```
adb pull /sdcard/screen.png
```

#### 截图：选项 2（更快）

执行以下一行命令：

(Marshmallow 及更早版本) :

```
adb shell screencap -p | perl -pe 's/ \x0D\x0A/g' > screen.png
```

(Nougat 及更高版本) :

```
adb shell screencap -p > screen.png
```

-p 标志将 screencap 命令的输出重定向到标准输出。管道传输的 Perl 表达式用于清理 Marshmallow 及更早版本中的一些行尾问题。然后，数据流被写入当前目录下名为 screen.png 的文件。更多信息请参见 [this article](#) 和 [this article](#)。

#### 视频

此功能仅在 KitKat 版本及通过 ADB 使用时有效。低于 KitKat 版本无法使用。要开始录制设备's 屏幕，请运行以下命令：

```
adb shell screenrecord /sdcard/example.mp4
```

该命令将使用默认设置开始录制设备's 屏幕，并将生成的视频保存到设备上的 /sdcard/example.mp4 文件中。

录制完成后，在命令提示符窗口按 Ctrl+C (Linux 中为 z) 停止屏幕录制。然后，您可以在指定的位置找到录制文件。请注意，屏幕录制文件保存在设备's 内部存储中，而非计算机上。

默认设置为使用设备's 标准屏幕分辨率，以 4Mbps 的比特率编码视频，并将最大录制时间设置为 180 秒。有关可用命令行选项的更多信息，请运行以下命令：

adb shell screenrecord --help，该命令无需设备 root 即可使用。希望这对您有帮助。

## 第 264.4 节：从设备拉取（推送）文件

您可以通过执行以下命令从设备拉取（下载）文件：

```
adb pull <remote> <local>
```

例如：

```
adb pull /sdcard/ ~/
```

您也可以将文件从计算机推送（上传）到设备：

```
adb push <local> <remote>
```

例如：

```
adb push ~/image.jpg /sdcard/
```

从设备检索数据库的示例

```
adb pull /sdcard/screen.png
```

#### Screen shot:Option 2 (faster)

Execute the following one-liner:

(Marshmallow and earlier):

```
adb shell screencap -p | perl -pe 's/\x0D\x0A/\x0A/g' > screen.png
```

(Nougat and later):

```
adb shell screencap -p > screen.png
```

The -p flag redirects the output of the screencap command to stdout. The Perl expression this is piped into cleans up some end-of-line issues on Marshmallow and earlier. The stream is then written to a file named screen.png within the current directory. See [this article](#) and [this article](#) for more information.

#### Video

This only works in KitKat and via ADB only. This does not work below KitKat. To start recording your device's screen, run the following command:

```
adb shell screenrecord /sdcard/example.mp4
```

This command will start recording your device's screen using the default settings and save the resulting video to a file at /sdcard/example.mp4 on your device.

When you're done recording, press Ctrl+C (z in Linux) in the Command Prompt window to stop the screen recording. You can then find the screen recording file at the location you specified. Note that the screen recording is saved to your device's internal storage, not to your computer.

The default settings are to use your device's standard screen resolution, encode the video at a bitrate of 4Mbps, and set the maximum screen recording time to 180 seconds. For more information about the command-line options you can use, run the following command:

```
adb shell screenrecord --help
```

This works without rooting the device. Hope this helps.

## Section 264.4: Pull (push) files from (to) the device

You may pull (download) files from the device by executing the following command:

```
adb pull <remote> <local>
```

For example:

```
adb pull /sdcard/ ~/
```

You may also push (upload) files from your computer to the device:

```
adb push <local> <remote>
```

For example:

```
adb push ~/image.jpg /sdcard/
```

Example to Retrieve Database from device

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name /databases/DATABASE_NAME
> /sdcard/file
```

## 第264.5节：打印已连接设备的详细列表

要获取所有连接到adb的设备的详细列表，请在终端中输入以下命令：

```
adb devices -l
```

### 示例输出

#### 连接设备列表

ZX1G425DC6	设备 usb:336592896X 产品:shamu 型号:Nexus_6 设备:shamu
013e4e127e59a868	设备 usb:337641472X 产品:bullhead 型号:Nexus_5X 设备:bullhead
ZX1D229KCN	设备 usb:335592811X 产品:titan_retdt 型号:XT1068 设备:titan_umtsds
A50PL	设备 usb:331592812X

- 第一列是设备的序列号。如果以emulator-开头，则该设备是模拟器。
- usb: 设备在USB子系统中的路径。
- product: 设备的产品代码。这非常依赖制造商，正如上面Archos设备A50PL的例子所示，可能为空。
- model: 设备型号。与product类似，也可以为空。
- device: 设备代码。这同样非常依赖制造商，也可以为空。

## 第264.6节：查看logcat

你可以将logcat作为adb命令运行，或者直接在模拟器或连接设备的shell提示符中运行。要使用adb查看日志输出，进入SDK platform-tools/目录并执行：

```
$ adb logcat
```

或者，你可以创建一个设备的shell连接，然后执行：

```
$ adb shell
$ logcat
```

一个有用的命令是：

```
adb logcat -v threadtime
```

这会以长消息格式显示日期、调用时间、优先级、标签，以及发出消息的线程的PID和TID。

### 过滤

Logcat 日志有所谓的日志级别：

V — 详细 (Verbose) , D — 调试 (Debug) , I — 信息 (Info) , W — 警告 (Warning) , E — 错误 (Error) , F — 致命 (Fatal) , S — 静默 (Silent)

你也可以按日志级别过滤 logcat。例如，如果你只想输出调试级别：

```
adb logcat *:D
```

Logcat 当然也可以按包名过滤，你也可以将其与日志级别过滤结合使用：

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name /databases/DATABASE_NAME
> /sdcard/file
```

## Section 264.5: Print verbose list of connected devices

To get a verbose list of all devices connected to adb, write the following command in your terminal:

```
adb devices -l
```

### Example Output

#### List of devices attached

ZX1G425DC6	device usb:336592896X product:shamu model:Nexus_6 device:shamu
013e4e127e59a868	device usb:337641472X product:bullhead model:Nexus_5X device:bullhead
ZX1D229KCN	device usb:335592811X product:titan_retdt model:XT1068 device:titan_umtsds
A50PL	device usb:331592812X

- The first column is the serial number of the device. If it starts with emulator-, this device is an emulator.
- usb: the path of the device in the USB subsystem.
- product : the product code of the device. This is very manufacturer-specific, and as you can see in the case of the Archos device A50PL above, it can be blank.
- model: the device model. Like product, can be empty.
- device : the device code. This is also very manufacturer-specific, and can be empty.

## Section 264.6: View logcat

You can run logcat as an adb command or directly in a shell prompt of your emulator or connected device. To view log output using adb, navigate to your SDK platform-tools/ directory and execute:

```
$ adb logcat
```

Alternatively, you can create a shell connection to a device and then execute:

```
$ adb shell
$ logcat
```

One useful command is:

```
adb logcat -v threadtime
```

This displays the date, invocation time, priority, tag, and the PID and TID of the thread issuing the message in a long message format.

### Filtering

Logcat logs got so called log levels:

V — Verbose, D — Debug, I — Info, W — Warning, E — Error, F — Fatal, S — Silent

You can filter logcat by log level as well. For instance if you want only to output Debug level:

```
adb logcat *:D
```

Logcat can be filtered by a package name, of course you can combine it with the log level filter:

```
adb logcat <package-name>:<log level>
```

你也可以使用 grep 过滤日志 (关于过滤 logcat 输出的更多内容见此处) :

```
adb logcat | grep <some text>
```

在 Windows 中, 可以使用 findstr 进行过滤, 例如:

```
adb logcat | findstr <some text>
```

要查看其他日志缓冲区 [main|events|radio], 请使用带有 -b 选项的 logcat 命令:

```
adb logcat -b radio
```

将输出保存到文件:

```
adb logcat > logcat.txt
```

保存输出到文件的同时也进行查看:

```
adb logcat | tee logcat.txt
```

清理日志:

```
adb logcat -c
```

## 第264.7节：查看并拉取应用的缓存文件

您可以使用此命令列出您自己的可调试apk的文件:

```
adb shell run-as <sample.package.id> ls /data/data/sample.package.id/cache
```

使用此脚本从缓存中拉取文件, 该脚本先将内容复制到sd卡, 拉取后再删除:

```
#!/bin/sh
adb shell "run-as <sample.package.id> cat '/data/data/<sample.package.id>/$1' > '/sdcard/$1'"
adb pull "/sdcard/$1"
adb shell "rm '/sdcard/$1'"
```

然后您可以这样从缓存中拉取文件:

```
./pull.sh cache/someCachedData.txt
```

### 通过ADB获取数据库文件

```
sudo adb -d shell "run-as com.example.name cat /data/data/com.example.name
/databases/STUDENT_DATABASE > /sdcard/file"
```

## 第264.8节：清除应用数据

可以使用adb清除特定应用的用户数据:

```
adb shell pm clear <package>
```

```
adb logcat <package-name>:<log level>
```

You can also filter the log using grep (more on filtering logcat output here):

```
adb logcat | grep <some text>
```

In Windows, filter can be used using findstr, for example:

```
adb logcat | findstr <some text>
```

To view alternative log buffer [main|events|radio], run the logcat with the -b option:

```
adb logcat -b radio
```

Save output in file:

```
adb logcat > logcat.txt
```

Save output in file while also watching it:

```
adb logcat | tee logcat.txt
```

Cleaning the logs:

```
adb logcat -c
```

## Section 264.7: View and pull cache files of an app

You may use this command for listing the files for your own debuggable apk:

```
adb shell run-as <sample.package.id> ls /data/data/sample.package.id/cache
```

And this script for pulling from cache, this copy the content to sdcard first, pull and then remove it at the end:

```
#!/bin/sh
adb shell "run-as <sample.package.id> cat '/data/data/<sample.package.id>/$1' > '/sdcard/$1'"
adb pull "/sdcard/$1"
adb shell "rm '/sdcard/$1'"
```

Then you can pull a file from cache like this:

```
./pull.sh cache/someCachedData.txt
```

### Get Database file via ADB

```
sudo adb -d shell "run-as com.example.name cat /data/data/com.example.name
/databases/STUDENT_DATABASE > /sdcard/file"
```

## Section 264.8: Clear application data

One can clear the user data of a specific app using adb:

```
adb shell pm clear <package>
```

这与在手机设置中浏览，选择应用并点击清除数据按钮的操作相同。

- pm 调用设备上的包管理器
- clear 删除与某个包相关的所有数据

## 第264.9节：查看设备上应用的内部数据 (data/data/<sample.package.id>)

首先，确保你的应用在AndroidManifest.xml中可以备份，即android:allowBackup 不是false。

备份命令：

```
adb -s <device_id> 备份 -noapk <sample.package.id>
```

使用 dd 命令创建 tar 文件：

```
dd if=backup.ab bs=1 skip=24 | python -c "import zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))" > backup.tar
```

解压 tar 文件：

```
tar -xvf backup.tar
```

然后您可以查看解压后的内容。

## 第 264.10 节：安装并运行应用程序

要安装 APK 文件，请使用以下命令：

```
adb install path/to/apk/file.apk
```

或者如果应用程序已存在且我们想重新安装

```
adb install -r 路径/到/apk/文件.apk
```

要卸载应用程序，必须指定其包名

```
adb uninstall 应用程序.包名.名称
```

使用以下命令启动具有指定包名（或应用中特定活动）的应用程序：

```
adb shell am start -n adb shell am start <包名>/<活动>
```

例如，要启动 Waze：

```
adb shell am start -n adb shell am start com.waze/com.waze.FreeMapAppActivity
```

## 第264.11节：发送广播

可以使用 adb 向 BroadcastReceiver 发送广播。

在此示例中，我们发送带有动作 com.test.app.ACTION 和字符串额外参数的广播，参数包中  
'foo'='bar'：

This is the same as to browse the settings on the phone, select the app and press on the clear data button.

- pm invokes the package manager on the device
- clear deletes all data associated with a package

## Section 264.9: View an app's internal data (data/data/<sample.package.id>) on a device

First, make sure your app can be backed up in AndroidManifest.xml, i.e. android:allowBackup is not **false**.

Backup command:

```
adb -s <device_id> backup -noapk <sample.package.id>
```

Create a tar with dd command:

```
dd if=backup.ab bs=1 skip=24 | python -c "import zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))" > backup.tar
```

Extract the tar:

```
tar -xvf backup.tar
```

You may then view the extracted content.

## Section 264.10: Install and run an application

To **install an APK file**, use the following command:

```
adb install path/to/apk/file.apk
```

or if the app is existing and we want to reinstall

```
adb install -r path/to/apk/file.apk
```

To **uninstall an application**, we have to specify its package

```
adb uninstall application.package.name
```

Use the following command to start an app with a provided package name (or a specific activity in an app):

```
adb shell am start -n adb shell am start <package>/<activity>
```

For example, to start Waze:

```
adb shell am start -n adb shell am start com.waze/com.waze.FreeMapAppActivity
```

## Section 264.11: Sending broadcast

It's possible to send broadcast to BroadcastReceiver with adb.

In this example we are sending broadcast with action com.test.app.ACTION and string extra in bundle  
'foo'='bar'：

```
adb shell am broadcast -a action com.test.app.ACTION --es foo "bar"
```

你可以将任何其他支持的类型放入bundle中，不仅限于字符串：

--ez	- 布尔值
--ei	- 整数
--el	- 长整型
--ef	- 浮点数
--eu	- URI
--eia	- 整数数组（用','分隔）
--ela	- 长整型数组（用','分隔）
--efa	- 浮点数组（以','分隔）
--esa	- 字符串数组（以','分隔）

要向特定包/类发送意图，可以使用-n或-p参数。

发送到包：

```
-p com.test.app
```

发送到特定组件（SomeReceiver类位于com.test.app包中）：

```
-n com.test.app/.SomeReceiver
```

有用的示例：

- 发送“启动完成”广播
- 通过adb命令设置时间后发送“时间更改”广播

## 第264.12节：备份

您可以使用adb backup命令备份您的设备。

```
adb backup [-f <文件>] [-apk|-noapk] [-obb|-noobb] [-shared|-noshared] [-all]
[-system|nosystem] [<包名...>]
```

**-f <文件名>** 指定文件名，默认：在当前目录创建 backup.ab

**-apk|noapk** 启用/禁用备份.apk文件本身，默认：-noapk

**-obb|noobb** 启用/禁用备份附加文件，默认：-noobb

**-shared|noshared** 备份设备的共享存储/SD卡内容，默认：-noshared

**-all** 备份所有已安装的应用程序

**-system|nosystem** 包含系统应用程序，默认：-system

**<包名>** 要备份的包列表（例如 com.example.android.myapp）（如果指定了-all，则不需要）

要进行完整设备备份，包括所有内容，请使用

```
adb backup -apk -obb -shared -all -system -f fullbackup.ab
```

```
adb shell am broadcast -a action com.test.app.ACTION --es foo "bar"
```

You can put any other supported type to bundle, not only strings:

--ez	- boolean
--ei	- integer
--el	- long
--ef	- float
--eu	- uri
--eia	- int array (separated by ',')
--ela	- long array (separated by ',')
--efa	- float array (separated by ',')
--esa	- string array (separated by ',')

To send intent to specific package/class -n or -p parameter can be used.

Sending to package:

```
-p com.test.app
```

Sending to a specific component (SomeReceiver class in com.test.app package):

```
-n com.test.app/.SomeReceiver
```

Useful examples:

- Sending a "boot complete" broadcast
- Sending a "time changed" broadcast after setting time via adb command

## Section 264.12: Backup

You can use the adb backup command to backup your device.

```
adb backup [-f <file>] [-apk|-noapk] [-obb|-noobb] [-shared|-noshared] [-all]
[-system|nosystem] [<packages...>]
```

**-f <filename>** specify filename **default**: creates backup.ab in the current directory

**-apk|noapk** enable/disable backup of .apks themself **default**: -noapk

**-obb|noobb** enable/disable backup of additional files **default**: -noobb

**-shared|noshared** backup device's shared storage / SD card contents **default**: -noshared

**-all** backup all installed applications

**-system|nosystem** include system applications **default**: -system

**<packages>** a list of packages to be backed up (e.g. com.example.android.myapp) (not needed if -all is specified)

For a full device backup, including everything, use

```
adb backup -apk -obb -shared -all -system -f fullbackup.ab
```

注意：进行完整备份可能需要较长时间。

要恢复备份，请使用

adb 恢复备份。ab

## 第 264.13 节：查看可用设备

命令：

adb 设备

结果示例：

```
设备列表
emulator-5554 设备
PhoneRT45Fr54 离线
123.454.67.45 无设备
```

第一列 - 设备序列号

第二列 - 连接状态

[Android 文档](#)

## 第 264.14 节：通过 IP 连接设备

在 Android 设备终端中输入以下命令

```
su
setprop service.adb.tcp.port 5555
stop adbd
启动 adbd
```

之后，你可以使用CMD和ADB通过以下命令进行连接

adb connect 192.168.0.101.5555

你也可以禁用它，并通过以下命令让ADB恢复监听USB

```
setprop service.adb.tcp.port -1
stop adbd
启动 adbd
```

从电脑端，如果你已经有USB访问权限（无需root）切换到使用Wi-Fi会

更简单。如果你已经通过USB连接设备，在电脑的命令行中输入以下命令

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

将192.168.0.101替换为设备IP

**Note:** Doing a full backup can take a long time.

In order to restore a backup, use

adb restore backup.ab

## Section 264.13: View available devices

Command:

adb devices

Result example:

```
List of devices attached
emulator-5554 device
PhoneRT45Fr54 offline
123.454.67.45 no device
```

First column - device serial number

Second column - connection status

[Android documentation](#)

## Section 264.14: Connect device by IP

Enter these commands in Android device [Terminal](#)

```
su
setprop service.adb.tcp.port 5555
stop adbd
start adbd
```

After this, you can use **CMD** and **ADB** to connect using the following command

adb connect 192.168.0.101.5555

And you can disable it and return ADB to listening on USB with

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

From a computer, if you have USB access already (no root required)

It is even easier to switch to using Wi-Fi, if you already have USB. From a command line on the computer that has the device connected via USB, issue the commands

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

Replace 192.168.0.101 with device IP

## 第264.15节：在Linux系统上安装ADB

如何使用你发行版的

通过apt安装到Ubuntu/Debian系统：

```
sudo apt-get update
sudo apt-get install adb
```

通过 yum 安装到 Fedora/CentOS 系统：

```
sudo yum check-update
sudo yum install android-tools
```

安装到Gentoo系统，使用portage：

```
sudo emerge --ask dev-util/android-tools
```

安装到openSUSE系统，使用zypper：

```
sudo zypper refresh
sudo zypper install android-tools
```

安装到Arch系统，使用pacman：

```
sudo pacman -Syyu
sudo pacman -S android-tools
```

## 第264.16节：查看活动栈

```
adb -s <serialNumber> shell dumpsys activity activities
```

与watch unix命令一起使用时非常有用：

```
watch -n 5 "adb -s <serialNumber> shell dumpsys activity activities | sed -En -e '/Stack #/p' -e '/Running activities,/Run #0/p'"
```

## 第264.17节：重启设备

您可以通过执行以下命令重启设备：

```
adb reboot
```

执行此命令以重启到引导加载程序：

```
adb reboot bootloader
```

重启到恢复模式：

```
adb reboot recovery
```

请注意，设备不会先关机！

## Section 264.15: Install ADB on Linux system

How to install the Android Debugging Bridge (ADB) to a Linux system with the terminal using your distro's repositories.

Install to Ubuntu/Debian system via apt:

```
sudo apt-get update
sudo apt-get install adb
```

Install to Fedora/CentOS system via yum:

```
sudo yum check-update
sudo yum install android-tools
```

Install to Gentoo system with portage:

```
sudo emerge --ask dev-util/android-tools
```

Install to openSUSE system with zypper:

```
sudo zypper refresh
sudo zypper install android-tools
```

Install to Arch system with pacman:

```
sudo pacman -Syyu
sudo pacman -S android-tools
```

## Section 264.16: View activity stack

```
adb -s <serialNumber> shell dumpsys activity activities
```

Very useful when used together with the watch unix command:

```
watch -n 5 "adb -s <serialNumber> shell dumpsys activity activities | sed -En -e '/Stack #/p' -e '/Running activities,/Run #0/p'"
```

## Section 264.17: Reboot device

You can reboot your device by executing the following command:

```
adb reboot
```

Perform this command to reboot into bootloader:

```
adb reboot bootloader
```

Reboot to recovery mode:

```
adb reboot recovery
```

Be aware that the device won't shutdown first!

## 第264.18节：读取设备信息

在终端中输入以下命令：

```
adb shell getprop
```

这将以键/值对的形式打印所有可用信息。

您可以通过在命令后附加特定键的名称来读取特定信息。例如：

```
adb shell getprop ro.product.model
```

以下是您可以获取的一些有趣信息：

- `ro.product.model`: 设备型号名称 (例如 Nexus 6P)
- `ro.build.version.sdk`: 设备的 API 级别 (例如 23)
- `ro.product.brand`: 设备品牌 (例如 Samsung)

完整示例输出

## 第 264.19 节：列出 Android 6.0 上需要用户运行时授权的所有权限

```
adb shell pm list permissions -g -d
```

## 第 264.20 节：打开/关闭 Wifi

打开：

```
adb shell svc wifi enable
```

关闭：

```
adb shell svc wifi disable
```

## 第264.21节：启动/停止 adb

启动 ADB：

```
adb kill-server
```

停止 ADB：

```
adb start-server
```

## Section 264.18: Read device information

Write the following command in your terminal:

```
adb shell getprop
```

This will print all available information in the form of key/value pairs.

You can just read specific information by appending the name of a specific key to the command. For example:

```
adb shell getprop ro.product.model
```

Here are a few interesting pieces of information that you can get:

- `ro.product.model`: Model name of the device (e.g. Nexus 6P)
- `ro.build.version.sdk`: API Level of the device (e.g. 23)
- `ro.product.brand`: Branding of the device (e.g. Samsung)

Full Example Output

## Section 264.19: List all permissions that require runtime grant from users on Android 6.0

```
adb shell pm list permissions -g -d
```

## Section 264.20: Turn on/off Wifi

Turn on:

```
adb shell svc wifi enable
```

Turn off:

```
adb shell svc wifi disable
```

## Section 264.21: Start/stop adb

Start ADB:

```
adb kill-server
```

Stop ADB:

```
adb start-server
```

# 第265章：Android 中使用资源进行本地化

## 第265.1节：“res”目录下每个文件夹的配置类型和限定符名称

在res文件夹下的每个资源目录（如上例所示）都可以有不同变体，这些变体以类似名称的目录存在，并在每个配置-类型后缀不同的限定符-值。

在我们的安卓项目中常见的带有不同限定符值后缀的``目录变体示例：

- drawable/
- drawable-en/
- drawable-fr-rCA/
- drawable-en-port/
- drawable-en-notouch-12key/
- drawable-port-ldpi/
- drawable-port-notouch-12key/

安卓资源所有不同配置类型及其限定符值的详尽列表：

	配置	限定符值
MCC 和 MNC	示例：	mcc310
		mcc310-mnc004
		mcc208-mnc00
	等	
语言和地区	示例：	en
		fr
		en-rUS
		fr-rFR
		fr-rCA
布局方向	ldrtl	
	ldltr	
最小宽度	swdp	
	示例：	sw320dp
		sw600dp
		sw720dp
可用宽度	wdp	
	w720dp	
	w1024dp	
可用高度	hdp	
	h720dp	
	h1024dp	
屏幕尺寸	小	

# Chapter 265: Localization with resources in Android

## Section 265.1: Configuration types and qualifier names for each folder under the "res" directory

Each resource directory under the res folder (listed in the example above) can have different variations of the contained resources in similarly named directory suffixed with different qualifier-values for each configuration-type.

Example of variations of `` directory with different qualifier values suffixed which are often seen in our android projects:

- drawable/
- drawable-en/
- drawable-fr-rCA/
- drawable-en-port/
- drawable-en-notouch-12key/
- drawable-port-ldpi/
- drawable-port-notouch-12key/

Exhaustive list of all different configuration types and their qualifier values for android resources:

	Configuration	Qualifier Values
MCC and MNC	Examples:	mcc310 mcc310-mnc004 mcc208-mnc00 etc.
Language and region	Examples:	en fr en-rUS fr-rFR fr-rCA
Layout Direction	ldrtl ldltr	
smallestWidth	swdp	Examples: sw320dp sw600dp sw720dp
Available width	wdp w720dp w1024dp	sw320dp sw600dp sw720dp
Available height	hdp h720dp h1024dp	hdp h720dp h1024dp
Screen size	small	small

正常	normal
大	large
超大	xlarge
屏幕比例	<b>Screen aspect</b>
长	long
不长	notlong
圆形屏幕	<b>Round screen</b>
圆形	round
不圆形	notround
屏幕方向	<b>Screen orientation</b>
竖屏	port
横屏	land
用户界面模式	<b>UI mode</b>
汽车	car
桌面	desk
电视	television
家电手表	appliancewatch
夜间模式	<b>Night mode</b>
夜晚	night
非夜晚	notnight
屏幕像素密度 (dpi)	<b>Screen pixel density (dpi)</b>
低密度屏幕 (ldpi)	ldpi
中密度屏幕 (mdpi)	mdpi
高密度屏幕 (hdpi)	hdpi
超高密度屏幕 (xhdpi)	xhdpi
超超高密度屏幕 (xxhdpi)	xxhdpi
超超超高密度屏幕 (xxxhdpi)	xxxhdpi
无密度屏幕 (nodpi)	nodpi
低分辨率屏幕 (tvdpi)	tvdpi
任意密度屏幕 (anydpi)	anydpi
触摸屏类型	<b>Touchscreen type</b>
无触摸 (notouch)	notouch
键盘可用性	<b>Keyboard availability</b>
手指触摸 (finger)	finger
物理键盘暴露 (keysexposed)	keysexposed
物理键盘隐藏 (keyshidden)	keyshidden
软键盘 (keyssoft)	keyssoft
主要文本输入方式	<b>Primary text input method</b>
无键盘 (nokeys)	nokeys
QWERTY键盘 (qwerty)	qwerty
导航键可用性	<b>Navigation key availability</b>
12键键盘 (12key)	12key
navexposed	navexposed
navhidden	navhidden
主要非触控导航方式 nonav	<b>Primary non-touch navigation method</b>
dpad	nonav
trackball	dpad
wheel	trackball
示例：	wheel
v3	
v4	
v7	
平台版本 (API 级别)	<b>Platform Version (API level)</b>
示例：	Examples:
v3	v3
v4	v4
v7	v7

## 第 265.2 节：为您的 Android 应用添加翻译

您必须为每种新语言创建不同的strings.xml文件。

## Section 265.2: Adding translation to your Android app

You have to create a different strings.xml file for every new language.

- 右键点击res文件夹
- 选择新建→值资源文件
- 从可用的限定符中选择一个区域设置
- 点击下一步按钮 (>>)
- 选择一种语言
- 命名文件strings.xml

#### strings.xml

```
<resources>
 <string name="app_name">测试应用程序</string>
 <string name="hello">你好，世界</string>
</resources>
```

#### strings.xml(印地语)

```
<resources>
 <string name="app_name">परीक्षण आवेदन</string>
 <string name="hello">नमस्ते दुनिया</string>
</resources>
```

#### 以编程方式设置语言：

```
public void setLocale(String locale) // 传入 "en", "hi" 等
{
 myLocale = new Locale(locale);
 // 将选定的语言保存到会话 - SharedPreferences.
 saveLocale(locale);
 // 更改语言环境。
 Locale.setDefault(myLocale);
 android.content.res.Configuration config = new android.content.res.Configuration();
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 config.setLocale(myLocale);
 } else {
 config.locale = myLocale;
 }
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
 getBaseContext().createConfigurationContext(config);
 } else {
 getBaseContext().getResources().updateConfiguration(config,
 getBaseContext().getResources().getDisplayMetrics());
 }
}
```

上述函数将更改引用自strings.xml的文本字段。例如，假设你有以下两个文本视图：

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/app_name" />
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/hello" />
```

然后，在更改区域设置后，具有IDapp\_name和hello的语言字符串将相应更改。

- Right-click on the *res* folder
- Choose *New → Values resource file*
- Select a locale from the available qualifiers
- Click on the *Next* button (>>)
- Select a language
- Name the file *strings.xml*

#### strings.xml

```
<resources>
 <string name="app_name">Testing Application</string>
 <string name="hello">Hello World</string>
</resources>
```

#### strings.xml(hi)

```
<resources>
 <string name="app_name">परीक्षण आवेदन</string>
 <string name="hello">नमस्ते दुनिया</string>
</resources>
```

#### Setting the language programmatically:

```
public void setLocale(String locale) // Pass "en", "hi", etc.
{
 myLocale = new Locale(locale);
 // Saving selected locale to session - SharedPreferences.
 saveLocale(locale);
 // Changing locale.
 Locale.setDefault(myLocale);
 android.content.res.Configuration config = new android.content.res.Configuration();
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 config.setLocale(myLocale);
 } else {
 config.locale = myLocale;
 }
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
 getBaseContext().createConfigurationContext(config);
 } else {
 getBaseContext().getResources().updateConfiguration(config,
 getBaseContext().getResources().getDisplayMetrics());
 }
}
```

The function above will change the text fields which are referenced from *strings.xml*. For example, assume that you have the following two text views:

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/app_name" />
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/hello" />
```

Then, after changing the locale, the language strings having the ids app\_name and hello will be changed accordingly.

## 第265.3节：“res”文件夹下资源目录的类型

在本地化时需要不同类型的资源，每种资源在Android项目结构中都有其对应的位置。以下是在res目录下可以放置的不同目录。每个目录中放置的资源类型在下表中进行了说明：

目录	资源类型
animator/ 定义属性动画的XML文件。	
anim/ 定义补间动画的XML文件。（属性动画也可以保存在此目录，但为了区分两种类型，属性动画更推荐放在 animator/目录。）	
color/ 定义颜色状态列表的XML文件。参见颜色状态列表资源 (Color State List Resource)	
drawable/ 子类型：位图文件-九宫格补丁（可重新调整大小的位图）-状态列表-形状-动画可绘制对象-其他可绘制对象-用于不同启动器图标密度的mipmap/可绘制文件。有关管理启动器图标的更多信息 请参见使用mipmap/文件夹，详见项目概述。	位图文件 (.png、.9.png、.jpg、.gif) 或编译成以下可绘制资源的XML文件 drawable/ subtypes: Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - mipmap/ 可绘制文件，有关管理启动器图标的更多信息 请参见使用mipmap/文件夹，详见项目概述。
layout/ 定义用户界面布局的XML文件。参见布局资源。	
menu/ 定义应用菜单的XML文件，如选项菜单、上下文菜单或子菜单。 参见菜单资源。	
raw/ 任意文件以原始形式保存。要使用原始输入流打开这些资源，请使用资源ID调用 Resources.openRawResource()，该ID为 R.raw.filename。 但是，如果您需要访问原始文件名和文件层级结构，您可以考虑将一些资源保存在 assets/ 目录中（而不是 res/raw/）。assets/ 中的文件没有资源ID，因此只能使用 AssetManager 来读取它们。	
values/ 包含简单值的 XML 文件，如字符串、整数和颜色，以及样式和主题	
xml/ 任意 XML 文件，可以通过调用 Resources.getXML() 在运行时读取。各种 XML 配置文件必须保存在这里，例如可搜索配置。	

## 第 265.4 节：以编程方式更改 Android 应用程序的语言环境

在上述示例中，您了解了如何本地化应用程序的资源。以下示例说明如何在应用程序内更改应用程序的语言环境，而不是从设备更改。为了仅更改应用程序语言环境，您可以使用以下语言环境工具。

```
import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.content.res.Resources;
import android.os.Build;
import android.preference.PreferenceManager;
import android.view.ContextThemeWrapper;

import java.util.Locale;

/**
 * 由 Umesh 于 2016/10/10 创建。
 */
public class LocaleUtils {

 private static Locale mLocale;

 public static void setLocale(Locale locale){
```

## Section 265.3: Type of resource directories under the "res" folder

When localizing different types of resources are required, each of which has its own home in the android project structure. Following are the different directories that we can place under the \res directory. The resource types placed in each of these directories are explained in the table below:

Directory	Resource Type
animator/ XML files that define property animations.	
anim/ XML files that define tween animations. (Property animations can also be saved in this directory, but the animator/ directory is preferred for property animations to distinguish between the two types.)	
color/ XML files that define a state list of colors. See Color State List Resource	
drawable/ subtypes: Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - mipmap/ Drawable files for different launcher icon densities. For more information on managing launcher icons with mipmap/ folders, see Managing Projects Overview.	"Bitmap files (.png, .9.png, .jpg, .gif) or XML files that are compiled into the following drawable resource
layout/ XML files that define a user interface layout. See Layout Resource.	drawable/ subtypes: Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables -
menu/ XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. See Menu Resource.	
raw/ Arbitrary files to save in their raw form. To open these resources with a raw InputStream, call Resources.openRawResource() with the resource ID, which is R.raw.filename.	
	However, if you need access to original file names and file hierarchy, you might consider saving some resources in the assets/ directory (instead of res/raw/). Files in assets/ are not given a resource ID, so you can read them only using AssetManager.
values/ XML files that contain simple values, such as strings, integers, and colors, as well as styles and themes	
xml/ Arbitrary XML files that can be read at runtime by calling Resources.getXML(). Various XML configuration files must be saved here, such as a searchable configuration.	

## Section 265.4: Change locale of android application programmatically

In above examples you understand how to localize resources of application. Following example explain how to change the application locale within application, not from device. In order to change Application locale only, you can use below locale util.

```
import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.content.res.Resources;
import android.os.Build;
import android.preference.PreferenceManager;
import android.view.ContextThemeWrapper;

import java.util.Locale;

/**
 * Created by Umesh on 10/10/16.
 */
public class LocaleUtils {

 private static Locale mLocale;
```

```

mLocale = locale;
 if(mLocale != null){
 Locale.setDefault(mLocale);
 }
}

public static void updateConfiguration(ContextThemeWrapper wrapper){
 if(mLocale != null && Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1){
 Configuration configuration = new Configuration();
 configuration.setLocale(mLocale);
 wrapper.applyOverrideConfiguration(configuration);
 }
}

public static void updateConfiguration(Application application, Configuration configuration){
 if(mLocale != null && Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN_MR1){
 Configuration config = new Configuration(configuration);
 config.locale = mLocale;
 Resources res = application.getBaseContext().getResources();
 res.updateConfiguration(configuration, res.getDisplayMetrics());
 }
}

public static void updateConfiguration(Context context, String language, String country){
 Locale locale = new Locale(language,country);
 setLocale(locale);
 if(mLocale != null){
 Resources res = context.getResources();
 Configuration configuration = res.getConfiguration();
 configuration.locale = mLocale;
 res.updateConfiguration(configuration,res.getDisplayMetrics());
 }
}

public static String getPrefLangCode(Context context) {
 return PreferenceManager.getDefaultSharedPreferences(context).getString("lang_code", "en");
}

public static void setPrefLangCode(Context context, String mPrefLangCode) {

SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
 editor.putString("lang_code",mPrefLangCode);
editor.commit();
}

public static String getPrefCountryCode(Context context) {
 return
PreferenceManager.getDefaultSharedPreferences(context).getString("country_code", "US");
}

public static void setPrefCountryCode(Context context,String mPrefCountryCode) {

SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
 editor.putString("country_code",mPrefCountryCode);
editor.commit();
}
}

```

```

mLocale = locale;
 if(mLocale != null){
 Locale.setDefault(mLocale);
 }
}

public static void updateConfiguration(ContextThemeWrapper wrapper){
 if(mLocale != null && Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1){
 Configuration configuration = new Configuration();
 configuration.setLocale(mLocale);
 wrapper.applyOverrideConfiguration(configuration);
 }
}

public static void updateConfiguration(Application application, Configuration configuration){
 if(mLocale != null && Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN_MR1){
 Configuration config = new Configuration(configuration);
 config.locale = mLocale;
 Resources res = application.getBaseContext().getResources();
 res.updateConfiguration(configuration, res.getDisplayMetrics());
 }
}

public static void updateConfiguration(Context context, String language, String country){
 Locale locale = new Locale(language,country);
 setLocale(locale);
 if(mLocale != null){
 Resources res = context.getResources();
 Configuration configuration = res.getConfiguration();
 configuration.locale = mLocale;
 res.updateConfiguration(configuration,res.getDisplayMetrics());
 }
}

public static String getPrefLangCode(Context context) {
 return PreferenceManager.getDefaultSharedPreferences(context).getString("lang_code", "en");
}

public static void setPrefLangCode(Context context, String mPrefLangCode) {

SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
 editor.putString("lang_code",mPrefLangCode);
editor.commit();
}

public static String getPrefCountryCode(Context context) {
 return
PreferenceManager.getDefaultSharedPreferences(context).getString("country_code", "US");
}

public static void setPrefCountryCode(Context context,String mPrefCountryCode) {

SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
 editor.putString("country_code",mPrefCountryCode);
editor.commit();
}
}

```

从 Application 类初始化用户偏好的语言环境。

```
public class LocaleApp extends Application{

 @Override
 public void onCreate() {
 super.onCreate();

 LocaleUtils.setLocale(new Locale(LocaleUtils.getPrefLangCode(this),
 LocaleUtils.getPrefCountryCode(this)));
 LocaleUtils.updateConfiguration(this, getResources().getConfiguration());
 }
}
```

你还需要创建一个基础 Activity，并让所有其他 Activity 继承该 Activity，这样你就可以只在一个地方更改应用程序的语言环境，如下所示：

```
public abstract class LocalizationActivity extends AppCompatActivity {

 public LocalizationActivity() {
 LocaleUtils.updateConfiguration(this);
 }

 // 我们只重写 onCreate
 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 }
}
```

注意：始终在构造函数中初始化区域设置。

现在你可以按如下方式使用 LocalizationActivity。

```
public class MainActivity extends LocalizationActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

注意：当你以编程方式更改应用程序的区域设置时，需要重启你的活动以使区域设置更改生效。为了使此解决方案正常工作，你可以在应用启动时从共享偏好中使用区域设置，并在你的 Manifest.xml 中设置 android:name=".LocaleApp"。

有时 Lint 检查器会提示创建发布版本。要解决此类问题，请按照以下选项操作。

第一：

如果你只想禁用某些字符串的翻译，则在默认的 string.xml 中添加以下属性

Initialize locale that user preferred, from Application class.

```
public class LocaleApp extends Application{

 @Override
 public void onCreate() {
 super.onCreate();

 LocaleUtils.setLocale(new Locale(LocaleUtils.getPrefLangCode(this),
 LocaleUtils.getPrefCountryCode(this)));
 LocaleUtils.updateConfiguration(this, getResources().getConfiguration());
 }
}
```

You also need to create a base activity and extend this activity to all other activity so that you can change locale of application only one place as follows :

```
public abstract class LocalizationActivity extends AppCompatActivity {

 public LocalizationActivity() {
 LocaleUtils.updateConfiguration(this);
 }

 // We only override onCreate
 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 }
}
```

Note : Always initialize locale in constructor.

Now you can use LocalizationActivity as follow.

```
public class MainActivity extends LocalizationActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
}
```

Note: When you change locale of application programmatically, need to restart your activity to take the effect of locale change In order to work properly for this solution you and use locale from shared preferences on app startup you android:name=".LocaleApp" in your Manifest.xml.

Sometimes Lint checker prompt to create the release build. To solve such issue follow below options.

First:

If you want to disable translation for some strings only then add following attribute to default string.xml

```
<string name="developer" translatable="false">开发者名称</string>
```

## 第二：

忽略资源文件中所有缺失的翻译，添加以下属性。这是字符串文件中tools命名空间的ignore属性，格式如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources
 xmlns:tools="http://schemas.android.com/tools"
 tools:ignore="MissingTranslation" >
 http://stackoverflow.com/documentation/android/3345/localization-with-resources-in-android#
 <!-- 你的字符串放这里；现在不需要translatable属性 -->

</resources>
```

## 第三：

禁用不可翻译字符串的另一种方法

<http://tools.android.com/recent/non-translatablestrings>

如果你有很多不应翻译的资源，可以将它们放在名为donottranslate.xml的文件中，lint会将它们全部视为不可翻译的资源。

## 第四：

你也可以在资源文件中添加区域设置

```
<resources
 xmlns:tools="http://schemas.android.com/tools"
 tools:locale="en" tools:ignore="MissingTranslation">
```

您也可以在 app/build.gradle 中禁用 lint 的缺失翻译检查

```
lintOptions {
 disable 'MissingTranslation'
}
```

## 第265.5节：货币

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

```
<string name="developer" translatable="false">Developer Name</string>
```

## Second:

Ignore all missing translation from resource file add following attribute It's the ignore attribute of the tools namespace in your strings file, as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources
 xmlns:tools="http://schemas.android.com/tools"
 tools:ignore="MissingTranslation" >
 http://stackoverflow.com/documentation/android/3345/localization-with-resources-in-android#
 <!-- your strings here; no need now for the translatable attribute -->

</resources>
```

## Third:

Another way to disable non-translatable string

<http://tools.android.com/recent/non-translatablestrings>

If you have a lot of resources that should not be translated, you can place them in a file named donottranslate.xml and lint will consider all of them non-translatable resources.

## Fourth:

You can also add locale in resource file

```
<resources
 xmlns:tools="http://schemas.android.com/tools"
 tools:locale="en" tools:ignore="MissingTranslation">
```

You can also disable missing translation check for lint from app/build.gradle

```
lintOptions {
 disable 'MissingTranslation'
}
```

## Section 265.5: Currency

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

# 第266章：将越南语字符串转换为 英语字符串 Android

## 第266.1节：示例：

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

转换后：

"Le Minh Thoai 是越南人"

## 第266.2节：将越南语字符串转换为无音调字符串

```
public static String convert(String str) {
 str = str.replaceAll("à|á|ả|ã|â|ã|ă|ã|ă|ã|ă|ã|ă|ã|ă|ã", "a");
 str = str.replaceAll("è|é|ě|é|ě|é|ě|é|ě|é|ě|é|ě|ě", "e");
 str = str.replaceAll("í|í|í|í|í", "i");
 str = str.replaceAll("ò|ó|ö|ó|ô|ó|ö|ó|ö|ó|ö|ó|ö|ó|ö", "o");
 str = str.replaceAll("ù|ú|ü|ú|û|ú|ü|ú|ü|ú|ü|ú|ü", "u");
 str = str.replaceAll("ỳ|ý|ÿ|ý", "y");
 str = str.replaceAll("đ", "d");

 str = str.replaceAll("À|Á|Ã|À|Ã|Ã|À|Ã|Ã|À|Ã|Ã|À|Ã|Ã", "A");
 str = str.replaceAll("È|É|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È", "E");
 str = str.replaceAll("Í|Í|Í|Í|Í", "I");
 str = str.replaceAll("Ò|Ó|Ö|Ó|Ô|Ó|Ö|Ó|Ö|Ó|Ö|Ó|Ö|Ó|Ö", "O");
 str = str.replaceAll("Ù|Ú|Ü|Ú|Û|Ú|Ü|Ú|Ü|Ú|Ü|Ú|Ü", "U");
 str = str.replaceAll("Ỳ|Ý|Ÿ|Ý", "Y");
 str = str.replaceAll("Đ", "D");
 return str;
}
```

# Chapter 266: Convert vietnamese string to english string Android

## Section 266.1: example:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

converted:

"Le Minh Thoai la nguoi Viet Nam"

## Section 266.2: Chuyển chuỗi Tiếng Việt thành chuỗi không dấu

```
public static String convert(String str) {
 str = str.replaceAll("à|á|ả|ã|â|ã|ă|ã|ă|ã|ă|ã|ă|ã|ă|ã", "a");
 str = str.replaceAll("è|é|ě|é|ě|é|ě|é|ě|é|ě|é|ě|ě", "e");
 str = str.replaceAll("í|í|í|í|í", "i");
 str = str.replaceAll("ò|ó|ö|ó|ô|ó|ö|ó|ö|ó|ö|ó|ö|ó|ö", "o");
 str = str.replaceAll("ù|ú|ü|ú|û|ú|ü|ú|ü|ú|ü|ú|ü", "u");
 str = str.replaceAll("ỳ|ý|ÿ|ý", "y");
 str = str.replaceAll("đ", "d");

 str = str.replaceAll("À|Á|Ã|À|Ã|Ã|À|Ã|Ã|À|Ã|Ã|À|Ã|Ã", "A");
 str = str.replaceAll("È|É|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È|Ã|È", "E");
 str = str.replaceAll("Í|Í|Í|Í|Í", "I");
 str = str.replaceAll("Ò|Ó|Ö|Ó|Ô|Ó|Ö|Ó|Ö|Ó|Ö|Ó|Ö|Ó|Ö", "O");
 str = str.replaceAll("Ù|Ú|Ü|Ú|Û|Ú|Ü|Ú|Ü|Ú|Ü|Ú|Ü", "U");
 str = str.replaceAll("Ỳ|Ý|Ÿ|Ý", "Y");
 str = str.replaceAll("Đ", "D");
 return str;
}
```

# 致谢

非常感谢所有来自Stack Overflow Documentation的人员帮助提供这些内容，  
更多更改可以发送至[web@petercv.com](mailto:web@petercv.com)以发布或更新新内容

<a href="#">0xalihn</a>	第191章和第231章
<a href="#">1SStorm</a>	第266章
<a href="#">3VYZkz7t</a>	第205章
<a href="#">A</a>	第159章
<a href="#">A.A.</a>	第7章
<a href="#">a.ch.</a>	第8章
<a href="#">阿瓦兹·贾瓦利</a>	第231章
<a href="#">阿卜杜拉·阿拉比</a>	第41章
<a href="#">阿卜杜拉</a>	第97章
<a href="#">阿比</a>	第41章、第47章和第253章
<a href="#">阿比谢克·贾因</a>	第1章、第16章、第40章、第47章、第218章、第227章和第258章
<a href="#">abhishesh</a>	第18章、第235章和第262章
<a href="#">阿比拉什</a>	第16章
<a href="#">Ab</a>	第62章
<a href="#">adalPaRi</a>	第62章和第99章
<a href="#">亚当·拉特兹曼</a>	第78章和第263章
<a href="#">adao7000</a>	第45章、第78章和第264章
<a href="#">阿达什·阿肖克</a>	第10章
<a href="#">阿迪卡里·比什瓦什</a>	第33章、第196章和第212章
<a href="#">阿迪尔·赛亚德</a>	第63、119和228章
<a href="#">阿德南</a>	第51章和第96章
<a href="#">阿德里安·佩雷斯</a>	第13章
<a href="#">AesSedai101</a>	第8章
<a href="#">艾哈迈德·阿加扎德</a>	第40、44、47、53、57、84、120、156、205、250和264章
<a href="#">ahmadalibaloch</a>	第41章
<a href="#">阿吉特·辛格</a>	第125章
<a href="#">阿卡什·帕特尔</a>	第16章和第37章
<a href="#">阿凯什瓦尔·贾</a>	第248章
<a href="#">等人</a>	第239章
<a href="#">阿拉·埃丁·杰巴利</a>	第1章
<a href="#">alanv</a>	第83章和第160章
<a href="#">亚历克斯·G</a>	第74章
<a href="#">亚历山大·斯特凡诺维奇</a>	第1、8、37、43、83和163章
<a href="#">亚历克斯</a>	第59章
<a href="#">亚历克斯·博内尔</a>	第16章
<a href="#">亚历克斯·陈加兰</a>	第37和82章
<a href="#">亚历克斯·埃尔绍夫</a>	第77章
<a href="#">亚历克斯·沙利文</a>	第95章
<a href="#">亚历山大·奥普里斯尼克</a>	第143章
<a href="#">阿列克谢·波卢索夫</a>	第69、246和263章
<a href="#">阿里·谢拉法特</a>	第21章
<a href="#">阿曼·安古拉拉</a>	第131章
<a href="#">阿米特</a>	第42章
<a href="#">阿米特·塔卡尔</a>	第264章
<a href="#">阿莫德·戈克哈莱</a>	第24章
<a href="#">阿南德·辛格</a>	第61章
<a href="#">阿纳托利</a>	第16章

# Credits

Thank you greatly to all the people from Stack Overflow Documentation who helped provide this content,  
more changes can be sent to [web@petercv.com](mailto:web@petercv.com) for new content to be published or updated

<a href="#">0xalihn</a>	Chapters 191 and 231
<a href="#">1SStorm</a>	Chapter 266
<a href="#">3VYZkz7t</a>	Chapter 205
<a href="#">A</a>	Chapter 159
<a href="#">A.A.</a>	Chapter 7
<a href="#">a.ch.</a>	Chapter 8
<a href="#">Aawaz Gyawali</a>	Chapter 231
<a href="#">Abdallah Alaraby</a>	Chapter 41
<a href="#">Abdullah</a>	Chapter 97
<a href="#">abhi</a>	Chapters 41, 47 and 253
<a href="#">Abhishek Jain</a>	Chapters 1, 16, 40, 47, 218, 227 and 258
<a href="#">abhishesh</a>	Chapters 18, 235 and 262
<a href="#">Abilash</a>	Chapter 16
<a href="#">Ab</a>	Chapter 62
<a href="#">adalPaRi</a>	Chapters 62 and 99
<a href="#">Adam Ratzman</a>	Chapters 78 and 263
<a href="#">adao7000</a>	Chapters 45, 78 and 264
<a href="#">Adarsh Ashok</a>	Chapter 10
<a href="#">Adhikari Bishwash</a>	Chapters 33, 196 and 212
<a href="#">Adil Saiyad</a>	Chapters 63, 119 and 228
<a href="#">Adnan</a>	Chapters 51 and 96
<a href="#">Adrián Pérez</a>	Chapter 13
<a href="#">AesSedai101</a>	Chapter 8
<a href="#">Ahmad Aghazadeh</a>	Chapters 40, 44, 47, 53, 57, 84, 120, 156, 205, 250 and 264
<a href="#">ahmadalibaloch</a>	Chapter 41
<a href="#">Ajit Singh</a>	Chapter 125
<a href="#">Akash Patel</a>	Chapters 16 and 37
<a href="#">Akeshwar Jha</a>	Chapter 248
<a href="#">AL</a>	Chapter 239
<a href="#">Ala Eddine JEBALI</a>	Chapter 1
<a href="#">alanv</a>	Chapters 83 and 160
<a href="#">Aleks G</a>	Chapter 74
<a href="#">Aleksandar Stefanović</a>	Chapters 1, 8, 37, 43, 83 and 163
<a href="#">Alex</a>	Chapter 59
<a href="#">Alex Bonel</a>	Chapter 16
<a href="#">Alex Chengalan</a>	Chapters 37 and 82
<a href="#">Alex Ershov</a>	Chapter 77
<a href="#">Alex Sullivan</a>	Chapter 95
<a href="#">Alexander Oprisnik</a>	Chapter 143
<a href="#">alexey polusov</a>	Chapters 69, 246 and 263
<a href="#">Ali Sherafat</a>	Chapter 21
<a href="#">Aman Anguralla</a>	Chapter 131
<a href="#">Amit</a>	Chapter 42
<a href="#">Amit Thakkar</a>	Chapter 264
<a href="#">Amod Gokhale</a>	Chapter 24
<a href="#">Anand Singh</a>	Chapter 61
<a href="#">anatoli</a>	Chapter 16

阿纳克斯  
安德森·K  
安迪极客  
安德烈·帕金斯  
安德烈·T  
安德鲁·布鲁克  
安德鲁·费尔南德斯  
[AndroidMechanic](#)  
[AndroidRuntimeException](#)  
[AndyRoid](#)  
[Anggrayudi H](#)  
[Anirudh Sharma](#)  
[Anish Mittal](#)  
[Anita Kunjir](#)  
安基特·波普利  
安基特·夏尔马  
安库尔·阿加瓦尔  
[Anonsage](#)  
[anoo\\_radha](#)  
安东尼奥  
阿努普·库尔卡尔尼  
[anupam\\_kamble](#)  
[AnV](#)  
阿普尔夫·帕尔马尔  
[appersiano](#)  
[aquib23](#)  
阿尔纳夫·M.  
阿尔皮特·甘地  
阿尔斯·蒂尔瓦  
阿里安·纳贾菲  
阿希什·兰詹  
阿希什·拉特希  
[astuter](#)  
阿特夫·哈雷斯  
[athor](#)  
阿提夫·法鲁克  
[Aurasphere](#)  
[auval](#)  
阿维纳什·R  
斧头  
阿尤什·班萨尔  
[B001](#)  
坏现金  
巴拉拉姆·纳亚克  
包子  
巴伦德  
巴尔特克·利平斯基  
比娜  
本  
本·P.  
[ben75](#)  
贝托·卡尔达斯  
[Baralga维·亚马努里](#)  
第146章  
第96章  
第126、233、236和248章  
第253章  
第95章  
第1、41、50和78章  
第41章  
第3、40、41、47、61、250、263、264和265章  
第41、47、61、76、93、97和219章  
第96章  
第40章  
第16、37、41、47、131和264章  
第42章  
第32章  
第142章  
第47章  
第98章和第190章  
第121章  
第34章  
第40、41、56、61和262章  
第264章  
第72章  
第50章  
第1章  
第80章  
第96章  
第247章  
第89章  
第52章  
第82章  
第40章  
第250章  
第92、97、227和262章  
第201章  
第114章  
第75和134章  
第8和112章  
第1、2、41、42、47、131、205、258、263和264章  
第41章  
第41章  
第133、170和199章  
第160章  
第45、82和239章  
第19章  
第219章  
第17、28和264章  
第8、16、37、41、47和81章  
第4章  
第88章和第258章  
第42章  
第255章  
第259章  
第91章和第216章

[Anax](#)  
[Anderson K](#)  
[AndiGeeky](#)  
[Andre Perkins](#)  
[Andrei T](#)  
[Andrew Brooke](#)  
[Andrew Fernandes](#)  
[AndroidMechanic](#)  
[AndroidRuntimeException](#)  
[AndyRoid](#)  
[Anggrayudi H](#)  
[Anirudh Sharma](#)  
[Anish Mittal](#)  
[Anita Kunjir](#)  
[Ankit Popli](#)  
[Ankit Sharma](#)  
[Ankur Aggarwal](#)  
[Anonsage](#)  
[anoo\\_radha](#)  
[antonio](#)  
[Anup Kulkarni](#)  
[anupam\\_kamble](#)  
[AnV](#)  
[Apoory Parmar](#)  
[appersiano](#)  
[aquib23](#)  
[Arnav M.](#)  
[Arpit Gandhi](#)  
[Arth Tilva](#)  
[Aryan Najafi](#)  
[Ashish Ranjan](#)  
[Ashish Rathee](#)  
[astuter](#)  
[Atef Hares](#)  
[athor](#)  
[Atif Farrukh](#)  
[Aurasphere](#)  
[auval](#)  
[Avinash R](#)  
[Axe](#)  
[Ayush Bansal](#)  
[B001](#)  
[BadCash](#)  
[BalaramNayak](#)  
[baozi](#)  
[Barend](#)  
[Bartek Lipinski](#)  
[Beena](#)  
[Ben](#)  
[Ben P.](#)  
[ben75](#)  
[Beto Caldas](#)  
[Bhargavi Yamanuri](#)  
Chapter 146  
Chapter 96  
Chapters 126, 233, 236 and 248  
Chapter 253  
Chapter 95  
Chapters 1, 41, 50 and 78  
Chapter 41  
Chapters 3, 40, 41, 47, 61, 250, 263, 264 and 265  
Chapters 41, 47, 61, 76, 93, 97 and 219  
Chapter 96  
Chapter 40  
Chapters 16, 37, 41, 47, 131 and 264  
Chapter 42  
Chapter 32  
Chapter 142  
Chapter 47  
Chapters 98 and 190  
Chapter 121  
Chapter 34  
Chapters 40, 41, 56, 61 and 262  
Chapter 264  
Chapter 72  
Chapter 50  
Chapter 1  
Chapter 80  
Chapter 96  
Chapter 247  
Chapter 89  
Chapter 52  
Chapter 82  
Chapter 40  
Chapter 250  
Chapters 92, 97, 227 and 262  
Chapter 201  
Chapter 114  
Chapters 75 and 134  
Chapters 8 and 112  
Chapters 1, 2, 41, 42, 47, 131, 205, 258, 263 and 264  
Chapter 41  
Chapter 41  
Chapters 133, 170 and 199  
Chapter 160  
Chapters 45, 82 and 239  
Chapter 19  
Chapter 219  
Chapters 17, 28 and 264  
Chapters 8, 16, 37, 41, 47 and 81  
Chapter 4  
Chapters 88 and 258  
Chapter 42  
Chapter 255  
Chapter 259  
Chapters 91 and 216

[biddulph.r](#)  
[黑带](#)  
[闪电克雷格](#)  
[布伦德尔](#)  
[犯错的哲学家](#)  
[bpoiss](#)  
[布拉杰·布尚·辛格](#)  
[布伦登·克罗姆霍特](#)  
[bricklore](#)  
[BrickTop](#)  
[布莱恩](#)  
[布莱恩·布莱斯](#)  
[巴迪](#)  
[布拉克·戴](#)  
[布尔哈努丁·拉希德](#)  
[busradeniz](#)  
[卡贝萨斯](#)  
[凯克·奥利维拉](#)  
[卡尔·普尔](#)  
[卡洛斯](#)  
[卡洛斯·博劳](#)  
[carvaq](#)  
[CaseyB](#)  
[卡西奥·兰迪姆](#)  
[cdeange](#)  
[查鲁](#)  
[钦坦·索尼](#)  
[芯片](#)  
[奇拉格·索兰基](#)  
[胆固醇](#)  
[克里斯·斯特拉顿](#)  
[克里斯特林·帕尼尔](#)  
[Code.IT](#)  
[Code\\_Life](#)  
[冷火](#)  
[commonSenseCode](#)  
[埃里克上尉](#)  
[cricket\\_007](#)  
[dakshbhattacharjee](#)  
[戴尔](#)  
[达利娅·普拉斯尼卡尔](#)  
[达米安·科兹拉克](#)  
[丹](#)  
[丹·赫尔姆](#)  
[丹尼尔·凯弗](#)  
[丹尼尔·纽金特](#)  
[丹尼尔·W.](#)  
[丹尼尔·迪苏](#)  
[丹尼尔·塞加托](#)  
[大卫·阿盖尔·萨克](#)  
[大卫·张](#)

第38章  
第2章、第40章和第264章  
第43章  
第187章和第188章  
第81章和第147章  
第41章  
第18章  
第47章  
第69章  
第10章  
第9、16和18章  
第39章  
第40章和第57章  
第205章和第264章  
第101章  
第58章  
第112章  
第39章  
第65章和第206章  
第14章  
第43章和第242章  
第53章  
第96章和第108章  
第96章  
第47、88、110、111和219章  
第1、2、8、37、41、76、78、81、103、125、239和264章  
第229章  
第21、23和91章  
第16章  
第61章  
第264章  
第225、226和240章  
第34和41章  
第219章  
第41章  
第258章  
第239章  
第42章和第47章  
第37章、第227章和第243章  
第264章  
第38章和第41章  
第40章、第42章和第227章  
第14章、第96章和第113章  
第37章和第40章  
第41章  
第3、8、9、12、13、16、24、38、40、41、42、45、61、62、69、70、72、74、82、88、92、96、103、114、115、143、146、218、227、229、239、243、249、250、251、258和263章  
第174章  
第21、41和219章  
第1章  
第39章  
第264章

[biddulph.r](#)  
[Blackbelt](#)  
[BlitzKraig](#)  
[Blundell](#)  
[Blundering Philosopher](#)  
[bpoiss](#)  
[Braj Bhushan Singh](#)  
[Brenden Kromhout](#)  
[bricklore](#)  
[BrickTop](#)  
[Bryan](#)  
[Bryan Bryce](#)  
[Buddy](#)  
[Burak Day](#)  
[Burhanuddin Rashid](#)  
[busradeniz](#)  
[Cabezas](#)  
[Caique Oliveira](#)  
[Carl Poole](#)  
[Carlos](#)  
[Carlos Borau](#)  
[carvaq](#)  
[CaseyB](#)  
[Cassio Landim](#)  
[cdeange](#)  
[Charu□](#)  
[Chintan Soni](#)  
[Chip](#)  
[Chirag Solanki](#)  
[Chol](#)  
[Chris Stratton](#)  
[Christlin Panneer](#)  
[Code.IT](#)  
[Code\\_Life](#)  
[Cold Fire](#)  
[commonSenseCode](#)  
[CptEric](#)  
[cricket\\_007](#)  
[dakshbhattacharjee](#)  
[Dale](#)  
[Dalija Prasnikar](#)  
[Damian Kozlak](#)  
[Dan](#)  
[Dan Hulme](#)  
[Daniel Käfer](#)  
[Daniel Nugent](#)  
[Daniel W.](#)  
[DanielDiSu](#)  
[Daniele Segato](#)  
[David Argyle Thacker](#)  
[David Cheung](#)  
Chapter 38  
Chapters 2, 40 and 264  
Chapter 43  
Chapters 187 and 188  
Chapters 81 and 147  
Chapter 41  
Chapter 18  
Chapter 47  
Chapter 69  
Chapter 10  
Chapters 9, 16 and 18  
Chapter 39  
Chapters 40 and 57  
Chapters 205 and 264  
Chapter 101  
Chapter 58  
Chapter 112  
Chapter 39  
Chapters 65 and 206  
Chapter 14  
Chapters 43 and 242  
Chapter 53  
Chapters 96 and 108  
Chapter 96  
Chapters 47, 88, 110, 111 and 219  
Chapters 1, 2, 8, 37, 41, 76, 78, 81, 103, 125, 239 and 264  
Chapter 229  
Chapters 21, 23 and 91  
Chapter 16  
Chapter 61  
Chapter 264  
Chapters 225, 226 and 240  
Chapters 34 and 41  
Chapter 219  
Chapter 41  
Chapter 258  
Chapter 239  
Chapters 42 and 47  
Chapters 37, 227 and 243  
Chapter 264  
Chapters 38 and 41  
Chapters 40, 42 and 227  
Chapters 14, 96 and 113  
Chapters 37 and 40  
Chapter 41  
Chapters 3, 8, 9, 12, 13, 16, 24, 38, 40, 41, 42, 45, 61, 62, 69, 70, 72, 74, 82, 88, 92, 96, 103, 114, 115, 143, 146, 218, 227, 229, 239, 243, 249, 250, 251, 258 and 263  
Chapter 174  
Chapters 21, 41 and 219  
Chapter 1  
Chapter 39  
Chapter 264

<a href="#">大卫·梅登雅克</a>	第17章和第112章	<a href="#">David Medenjak</a>	Chapters 17 and 112
<a href="#">davidgiga1993</a>	第27章	<a href="#">davidgiga1993</a>	Chapter 27
<a href="#">DeKaNszn</a>	第220章	<a href="#">DeKaNszn</a>	Chapter 220
<a href="#">dev.mi</a>	第37章	<a href="#">dev.mi</a>	Chapter 37
<a href="#">devnull69</a>	第70章和第219章	<a href="#">devnull69</a>	Chapters 70 and 219
<a href="#">达瓦尔·索兰基</a>	第96章	<a href="#">Dhaval Solanki</a>	Chapter 96
<a href="#">迪玛·罗斯托皮拉</a>	第250章	<a href="#">Dima Rostopira</a>	Chapter 250
<a href="#">迪内什·乔杜里</a>	第73章和第111章	<a href="#">Dinesh Choudhary</a>	Chapters 73 and 111
<a href="#">磁盘破坏者</a>	第86章	<a href="#">Disk Crashers</a>	Chapter 86
<a href="#">Dmide</a>	第25章	<a href="#">Dmide</a>	Chapter 25
<a href="#">唐·查卡潘</a>	第77章	<a href="#">Don Chakkappan</a>	Chapter 77
<a href="#">多隆·贝哈尔</a>	第1章	<a href="#">Doron Behar</a>	Chapter 1
<a href="#">多隆·雅科夫列夫</a>	第55章和第185章	<a href="#">Doron Yakovlev</a>	Chapters 55 and 185
<a href="#">道格拉斯·德拉蒙德</a>	第7章	<a href="#">Douglas Drumond</a>	Chapter 7
<a href="#">drulabs</a>	第231章	<a href="#">drulabs</a>	Chapter 231
<a href="#">段·布雷桑</a>	第74章和第219章	<a href="#">Duan Bressan</a>	Chapters 74 and 219
<a href="#">达斯</a>	第69章和第238章	<a href="#">Dus</a>	Chapters 69 and 238
<a href="#">埃格·库祖巴西奥卢</a>	第241章	<a href="#">Ege Kuzubasioglu</a>	Chapter 241
<a href="#">Eixx</a>	第81章	<a href="#">Eixx</a>	Chapter 81
<a href="#">Ekin</a>	第125章	<a href="#">Ekin</a>	Chapter 125
<a href="#">EKN</a>	第41章和第131章	<a href="#">EKN</a>	Chapters 41 and 131
<a href="#">EmmanuelMess</a>	第135章	<a href="#">EmmanuelMess</a>	Chapter 135
<a href="#">Endzeit</a>	第98章	<a href="#">Endzeit</a>	Chapter 98
<a href="#">恩里克·德·米格尔</a>	第210章	<a href="#">Enrique de Miguel</a>	Chapter 210
<a href="#">EpicPandaForce</a>	第112、113和127章	<a href="#">EpicPandaForce</a>	Chapters 112, 113 and 127
<a href="#">工程师考希克·卡贾瓦达拉</a>	第14和195章	<a href="#">Er. Kaushik Kajavadar</a>	Chapters 14 and 195
<a href="#">埃里克·米纳里尼</a>	第1、41、42和57章	<a href="#">Erik Minarini</a>	Chapters 1, 41, 42 and 57
<a href="#">尤金·马尔蒂诺夫</a>	第184和263章	<a href="#">Eugen Martynov</a>	Chapters 184 and 263
<a href="#">法比安·坦普</a>	第250章	<a href="#">Fabian Tamp</a>	Chapter 250
<a href="#">法比奥</a>	第47、139、148、194、205、264和265章	<a href="#">Fabio</a>	Chapters 47, 139, 148, 194, 205, 264 and 265
<a href="#">法里德</a>	第204章	<a href="#">Farid</a>	Chapter 204
<a href="#">费利克斯·埃尔德曼</a>	第19章	<a href="#">Felix Edelmann</a>	Chapter 19
<a href="#">弗莱恩</a>	第59章	<a href="#">Flayn</a>	Chapter 59
<a href="#">弗洛恩</a>	第8、34、38、41、47和71章	<a href="#">Floern</a>	Chapters 8, 34, 38, 41, 47 and 71
<a href="#">弗洛朗·斯帕休</a>	第9、47和218章	<a href="#">Florent Spahiu</a>	Chapters 9, 47 and 218
<a href="#">飞行彭巴</a>	第158章	<a href="#">FlyingPumba</a>	Chapter 158
<a href="#">弗朗克·德尔农库尔</a>	第41章	<a href="#">Franck Dernoncourt</a>	Chapter 41
<a href="#">弗雷德·马吉斯基</a>	第137章和第250章	<a href="#">FredMaggiowski</a>	Chapters 137 and 250
<a href="#">弗里克·诺尔蒂尔</a>	第250章	<a href="#">Freek Nortier</a>	Chapter 250
<a href="#">来自第七天空</a>	第18章	<a href="#">FromTheSeventhSky</a>	Chapter 18
<a href="#">冻酸奶</a>	第142章	<a href="#">Froyo</a>	Chapter 142
<a href="#">福瓦内科</a>	第18章	<a href="#">fuwaneko</a>	Chapter 18
<a href="#">fyfone 谷歌</a>	第44章、第156章、第205章和第264章	<a href="#">fyfone Google</a>	Chapters 44, 156, 205 and 264
<a href="#">g4s8</a>	第24、28、34、41、42、72、130、187和264章	<a href="#">g4s8</a>	Chapters 24, 28, 34, 41, 42, 72, 130, 187 and 264
<a href="#">gaara87</a>	第4、39、163、172和176章	<a href="#">gaara87</a>	Chapters 4, 39, 163, 172 and 176
<a href="#">盖布·塞昌</a>	第262章	<a href="#">Gabe Sechan</a>	Chapter 262
<a href="#">加布里埃莱·马里奥蒂</a>	第1、7、8、9、10、11、13、14、15、16、17、19、26、37、40、47、50、61、66、76、78、83、88、92、98、103、117、118、132、144、153、154、218、219、227、229、230、231、233、244、250、251、252、255、258和264章	<a href="#">Gabriele Mariotti</a>	Chapters 1, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 26, 37, 40, 47, 50, 61, 66, 76, 78, 83, 88, 92, 98, 103, 117, 118, 132, 144, 153, 154, 218, 219, 227, 229, 230, 231, 233, 244, 250, 251, 252, 255, 258 and 264
<a href="#">Gaket</a>	第1章	<a href="#">Gaket</a>	Chapter 1
<a href="#">加尔·耶迪多维奇</a>	第37章	<a href="#">Gal Yedidovich</a>	Chapter 37
<a href="#">gattsbr</a>	第263章	<a href="#">gattsbr</a>	Chapter 263
<a href="#">高拉夫·金达尔</a>	第8章	<a href="#">Gaurav Jindal</a>	Chapter 8

[gbansal](#) 第40章、第69章和第126章  
[盖尔特](#)  
[GensaGames](#) 第16章和第255章  
[杰拉德](#)  
[甘希亚姆·夏尔马](#)  
[吉安·帕特里克·昆塔纳](#)  
[扬尼斯](#)  
[吉南迪](#)  
[戈坎·阿里克](#)  
[戈尔格](#)  
[格雷厄姆·史密斯](#)  
[grebulon](#)  
[格雷格·T](#)  
[吉列尔梅·托雷斯·卡斯特罗](#)  
[吉约姆·安贝尔](#)  
[吉列尔莫·加西亚](#)  
[GurpreetSK95](#)  
[gus27](#)  
[H. Pauwelyn](#)  
[h22](#)  
[Hamed Gh](#)  
[Hamed Momeni](#)  
[hankide](#)  
[哈农·亚西尔](#)  
[哈里什·吉亚纳尼](#)  
[哈什·潘迪](#)  
[哈什·沙尔马](#)  
[哈西夫·赛义德](#)  
[HDehghani](#)  
[hello world](#)  
[herrmartell](#)  
[嗨，我是Frogatto](#)  
[希伦·帕特尔](#)  
[希特什·萨胡](#)  
[honk](#)  
[侯赛因·埃尔·费基](#)  
[Ic2h](#)  
[黑崎一护](#)  
[鱼人马](#)  
[iDevRoids](#)  
[伊利亚·布洛赫](#)  
[伊利亚·克罗尔](#)  
[伊曼·哈米迪](#)  
[因达德](#)  
[IncrediApp](#)  
[inetphantom](#)  
[失眠者](#)  
[因齐姆·塔里克 IT](#)  
[iravul](#)  
[伊尔凡·拉扎](#)

[gbansal](#) Chapters 40, 69 and 126  
[Geert](#) Chapter 100  
[GensaGames](#) Chapters 16 and 255  
[gerard](#) Chapter 218  
[Ghanshyam Sharma](#) Chapter 199  
[Gian Patrick Quintana](#) Chapter 209  
[Giannis](#) Chapter 235  
[Ginandi](#) Chapter 219  
[Gokhan Arik](#) Chapter 180  
[Gorg](#) Chapter 227  
[Graham Smith](#) Chapter 219  
[grebulon](#) Chapter 264  
[Greg T](#) Chapters 40, 67, 70, 96, 149, 160 and 262  
[Guilherme Torres Castro](#) Chapter 31  
[Guillaume Imbert](#) Chapter 39  
[Guillermo García](#) Chapter 37  
[GurpreetSK95](#) Chapter 176  
[gus27](#) Chapters 109 and 115  
[H. Pauwelyn](#) Chapter 114  
[h22](#) Chapter 48  
[Hamed Gh](#) Chapter 166  
[Hamed Momeni](#) Chapter 108  
[hankide](#) Chapters 1 and 47  
[Hannoun Yassir](#) Chapters 205 and 264  
[Harish Gyanani](#) Chapters 1, 5, 28, 38, 41, 99, 134, 201, 219 and 265  
[Harsh Pandey](#) Chapter 76  
[Harsh Sharma](#) Chapter 66  
[Hasif Seyd](#) Chapter 71  
[HDehghani](#) Chapter 34  
[hello world](#) Chapter 84  
[herrmartell](#) Chapter 98  
[Hi I'm Frogatto](#) Chapters 41, 42, 113, 137 and 250  
[Hiren Patel](#) Chapters 2, 4, 25, 34, 42, 53, 81, 82, 117, 118, 128, 141, 150, 152, 160 and 239  
[Hitesh Sahu](#) Chapter 155  
[honk](#) Chapters 4, 11, 18, 33, 37, 38, 44, 57, 58, 63, 74, 75, 86, 87, 91, 98, 100, 112, 116, 118, 119, 122, 133, 134, 139, 145, 149, 152, 155, 163, 165, 179, 194, 195, 200, 224, 228, 231, 233, 235, 238, 240, 242, 248, 252, 257, 258, 259, 264 and 265  
[Hussein El Feky](#) Chapter 102  
[Ic2h](#) Chapter 218  
[Ichigo Kurosaki](#) Chapters 37, 62, 126 and 250  
[Ichthyocentaur](#) Chapters 1, 47, 72, 92 and 179  
[iDevRoids](#) Chapter 261  
[Ilya Blokh](#) Chapter 227  
[Ilya Krol](#) Chapter 61  
[Iman Hamidi](#) Chapter 91  
[Imdad](#) Chapter 77  
[IncrediApp](#) Chapters 40 and 218  
[inetphantom](#) Chapter 1  
[insomniac](#) Chapter 41  
[Inzimam Tariq IT](#) Chapter 2  
[iravul](#) Chapters 22 and 152  
[Irfan Raza](#) Chapter 41

铁人  
伊山·费尔南多  
伊希塔·辛哈  
尤利安·波佩斯库  
伊万·伍尔  
jj  
j2ko  
雅各布  
贾加帕蒂  
贾格斯  
詹姆斯·帕森斯  
贾西姆·阿巴斯  
杰森·伯恩  
杰森·罗宾逊  
jasonlam604  
杰姆斯·比尔登  
让·维托尔  
吉特尔  
延斯V  
jgm  
jim  
吉内什·弗朗西斯  
吉特什·达尔萨尼亞  
Jl86  
jlynch630  
乔尔·格里特  
乔尔·普拉达  
约翰·斯诺  
johnrao07  
乔恩·亚当斯  
乔纳斯·科里茨  
JonasCz  
约斯特·弗布拉肯  
约旦  
乔迪·卡斯蒂利亚  
乔斯坎德鲁  
约书亚  
JoxTraex  
judepereira  
k3b  
kalan  
kann  
卡兰·纳格帕尔  
卡兰·拉兹丹  
KATHYxx  
考希克·NP  
凯万·N  
KDeogharkar  
凯达尔·滕多尔卡尔  
柯柳月  
凯文·迪特拉格利亚  
kld  
翠鸟福克

第16章、第41章和第99章  
第79章  
第37章、第60章和第83章  
第39章  
第41章  
第131章  
第24章  
第16章  
第237章  
第224章  
第43章  
第57章  
第125章  
第83章和第251章  
第24章  
第62章  
第41章  
第1章、第250章、第262章和第263章  
第107章  
第16章、第42章、第47章和第239章  
第264章  
第76章、第160章和第265章  
第227章  
第62章和第227章  
第37章  
第28章  
第250章  
第46章  
第34章和第85章  
第2、5、28、46、56、98、129、131、134和263章  
第256章  
第24、40、45和146章  
第49章  
第40和120章  
第43章  
第72章  
第16章  
第250章  
第203章  
第47章、第135章和第250章  
第31章  
第41章  
第41章  
第58章  
第96章  
第154章  
第16、40、41、57、66和141章  
第38和74章  
第15章  
第217和221章  
第16章  
第14、96和218章  
第258章

Ironman  
Ishan Fernando  
Ishita Sinha  
Iulian Popescu  
Ivan Wooll  
jj  
j2ko  
Jacob  
jagapathi  
Jaggs  
James Parsons  
Jaseem Abbas  
Jason Bourne  
Jason Robinson  
jasonlam604  
Jaymes Bearden  
Jean Vitor  
Jeeter  
JensV  
jgm  
jim  
Jinesh Francis  
Jitesh Dalsaniya  
Jl86  
jlynch630  
Joel Gritter  
Joel Prada  
John Snow  
johnrao07  
Jon Adams  
Jonas Köritz  
JonasCz  
Joost Verbraeken  
Jordan  
Jordi Castilla  
Joscandreu  
Joshua  
JoxTraex  
judepereira  
k3b  
kalan  
kann  
Karan Nagpal  
Karan Razdan  
KATHYxx  
Kaushik NP  
Kayvan N  
KDeogharkar  
Kedar Tendolkar  
KeLiuyue  
Kevin DiTraglia  
kld  
Kingfisher Phuoc

Chapters 16, 41 and 99  
Chapter 79  
Chapters 37, 60 and 83  
Chapter 39  
Chapter 41  
Chapter 131  
Chapter 24  
Chapter 16  
Chapter 237  
Chapter 224  
Chapter 43  
Chapter 57  
Chapter 125  
Chapters 83 and 251  
Chapter 24  
Chapter 62  
Chapter 41  
Chapters 1, 250, 262 and 263  
Chapter 107  
Chapters 16, 42, 47 and 239  
Chapter 264  
Chapters 76, 160 and 265  
Chapter 227  
Chapters 62 and 227  
Chapter 37  
Chapter 28  
Chapter 250  
Chapter 46  
Chapters 34 and 85  
Chapters 2, 5, 28, 46, 56, 98, 129, 131, 134 and 263  
Chapter 256  
Chapters 24, 40, 45 and 146  
Chapter 49  
Chapters 40 and 120  
Chapter 43  
Chapter 72  
Chapter 16  
Chapter 250  
Chapter 203  
Chapters 47, 135 and 250  
Chapter 31  
Chapter 41  
Chapter 41  
Chapter 58  
Chapter 96  
Chapter 154  
Chapters 16, 40, 41, 57, 66 and 141  
Chapters 38 and 74  
Chapter 15  
Chapters 217 and 221  
Chapter 16  
Chapters 14, 96 and 218  
Chapter 258

基兰·贝尼·约瑟夫  
基里尔·库拉科夫  
工具包  
克林格·克朗  
克诺索斯  
克里山  
克里希纳坎特  
kRiZ  
克鲁纳尔·帕特尔  
黑带 (KuroObi)  
L. 斯威夫特  
劳雷尔  
懒惰忍者  
利奥  
Leo.Han  
刘易斯·麦基里  
洛克什·德赛  
长枪骑士  
西迪厄斯勋爵  
卢卡斯·保利洛  
卢卡斯  
M D P  
M M  
马杜卡尔·赫巴尔  
马格什·潘迪安  
马哈茂德·易卜拉欣  
马莱克·希贾齐  
马诺斯  
马尔科·马尔基奥里  
马库斯·贝克尔  
MarGenDo  
马里奥·伦奇  
马克·奥梅舍  
马克·伊斯里  
棉花糖  
MashukKhan  
马塔斯·瓦伊特凯维休斯  
MathaN  
mattfred  
Mauker  
Max  
马克斯·麦金尼  
mayojava  
Medusalix  
Menasheh  
mhenryk  
迈克尔·艾伦  
迈克尔·斯皮茨因  
迈克尔·维斯科沃  
米歇尔  
MidasLefko  
米格尔·欣卡皮埃·C  
米卡尔·奥尔森

第1章和第145章  
第72章  
第118章  
第38章  
第151章  
第215章  
第207章  
第168章  
第64章和第229章  
第86章  
第219章  
第156章  
第72章和第227章  
第229章  
第72章  
第8章、第61章和第163章  
第186章和第260章  
第39章  
第40、78和178章  
第45、61和227章  
第78、120和176章  
第62和262章  
第244章  
第125章  
第134章  
第82章和第234章  
第144章  
第175章  
第40章和第84章  
第130章  
第182章  
第263章  
第41章  
第1章  
第76章  
第235章  
第1章  
第1章、第16章、第37章、第41章和第250章  
第112章  
第41章、第218章和第219章  
第37章、第41章、第61章、第97章和第131章  
第40章  
第46章  
第72章和第136章  
第38章、第46章和第264章  
第61章  
第1章  
第43章、第45章和第218章  
第251章  
第4章  
第19章和第82章  
第65章和第153章  
第258章

Kiran Benny Joseph  
Kirill Kulakov  
kit  
Kling Klang  
Knossos  
krishan  
Krishnakanth  
kRiZ  
krunal Patel  
KuroObi  
L. Swifter  
Laurel  
Lazy Ninja  
Leo  
Leo.Han  
Lewis McGeary  
Lokesh Desai  
Long Ranger  
LordSidious  
Lucas Paollo  
Lukas  
M D P  
M M  
Madhukar Hebbar  
Magesh Pandian  
Mahmoud Ibrahim  
Malek Hijazi  
Manos  
Marco Marchiori  
Marcus Becker  
MarGenDo  
Mario Lenci  
Mark Ormesher  
Mark Yisri  
marshmallow  
MashukKhan  
Matas Vaitkevicius  
MathaN  
mattfred  
Mauker  
Max  
Max McKinney  
mayojava  
Medusalix  
Menasheh  
mhenryk  
Michael Allan  
Michael Spitsin  
Michael Vescovo  
Michele  
MidasLefko  
MiguelHincapieC  
Mikael Ohlson

Chapters 1 and 145  
Chapter 72  
Chapter 118  
Chapter 38  
Chapter 151  
Chapter 215  
Chapter 207  
Chapter 168  
Chapters 64 and 229  
Chapter 86  
Chapter 219  
Chapter 156  
Chapters 72 and 227  
Chapter 229  
Chapter 72  
Chapters 8, 61 and 163  
Chapters 186 and 260  
Chapter 39  
Chapters 40, 78 and 178  
Chapters 45, 61 and 227  
Chapters 78, 120 and 176  
Chapters 62 and 262  
Chapter 244  
Chapter 125  
Chapter 134  
Chapters 82 and 234  
Chapter 144  
Chapter 175  
Chapters 40 and 84  
Chapter 130  
Chapter 182  
Chapter 263  
Chapter 41  
Chapter 1  
Chapter 76  
Chapter 235  
Chapter 1  
Chapters 1, 16, 37, 41 and 250  
Chapter 112  
Chapters 41, 218 and 219  
Chapters 37, 41, 61, 97 and 131  
Chapter 40  
Chapter 46  
Chapters 72 and 136  
Chapters 38, 46 and 264  
Chapter 61  
Chapter 1  
Chapters 43, 45 and 218  
Chapter 251  
Chapter 4  
Chapters 19 and 82  
Chapters 65 and 153  
Chapter 258

<a href="#">迈克</a>	第88章	<a href="#">Mike</a>	Chapter 88
<a href="#">迈克·拉伦</a>	第250章	<a href="#">Mike Laren</a>	Chapter 250
<a href="#">迈克·斯卡梅尔</a>	第71章和第76章	<a href="#">Mike Scamell</a>	Chapters 71 and 76
<a href="#">米拉德·努里</a>	第92章和第99章	<a href="#">Milad Nouri</a>	Chapters 92 and 99
<a href="#">米娜·萨米</a>	第239章	<a href="#">Mina Samy</a>	Chapter 239
<a href="#">小姐C</a>	第76章	<a href="#">miss C</a>	Chapter 76
<a href="#">mklimek</a>	第16章	<a href="#">mklimek</a>	Chapter 16
<a href="#">mmBs</a>	第37章和第97章	<a href="#">mmBs</a>	Chapters 37 and 97
<a href="#">莫查马德·陶菲克·希达亚特</a>	第62章和第76章	<a href="#">Mochamad Taufik Hidayat</a>	Chapters 62 and 76
<a href="#">莫尼什·坎布尔</a>	第218章	<a href="#">Monish Kamble</a>	Chapter 218
<a href="#">哲学硕士</a>	第227章	<a href="#">MPhil</a>	Chapter 227
<a href="#">mpkuth</a>	第31章、第37章、第78章、第117章、第138章和第265章	<a href="#">mpkuth</a>	Chapters 31, 37, 78, 117, 138 and 265
<a href="#">Mr.7</a>	第7章和第8章	<a href="#">Mr.7</a>	Chapters 7 and 8
<a href="#">萨尔蒙先生</a>	第78章	<a href="#">MrSalmon</a>	Chapter 78
<a href="#">穆图维宁</a>	第96、176和258章	<a href="#">mrtuovinen</a>	Chapters 96, 176 and 258
<a href="#">姆舒克拉</a>	第47章	<a href="#">mshukla</a>	Chapter 47
<a href="#">穆罕默德·乌迈尔</a>	第85章	<a href="#">Muhammad Umair Shafique</a>	Chapter 85
<a href="#">沙菲克</a>		<a href="#">Muhammad Younas</a>	Chapter 20
<a href="#">穆罕默德·尤纳斯</a>	第20章	<a href="#">Muhammed Refaat</a>	Chapters 41, 43, 71 and 218
<a href="#">穆罕默德·雷法特</a>	第41、43、71和218章	<a href="#">Mukesh Kumar Swami</a>	Chapter 101
<a href="#">穆凯什·库马尔·斯瓦米</a>	第101章	<a href="#">Murali</a>	Chapter 58
<a href="#">穆拉利</a>	第58章	<a href="#">Muthukrishnan Rajendran</a>	Chapters 17, 101, 137, 154 and 185
<a href="#">穆图克里希南·拉杰德兰</a>	第17、101、137、154和185章	<a href="#">Myon</a>	Chapter 56
<a href="#">迈昂</a>	第56章	<a href="#">N</a>	Chapter 106
<a href="#">N</a>	第106章	<a href="#">N</a>	Chapters 2, 41, 47, 218, 251 and 253
<a href="#">N</a>	第2、41、47、218、251和253章	<a href="#">Nambi</a>	Chapters 55 and 218
<a href="#">南比</a>	第55章和第218章	<a href="#">Namnodorel</a>	Chapter 253
<a href="#">Namnodorel</a>	第253章	<a href="#">Narayan Acharya</a>	Chapter 42
<a href="#">纳拉扬·阿查里亚</a>	第42章	<a href="#">narko</a>	Chapter 239
<a href="#">narko</a>	第239章	<a href="#">NashHorn</a>	Chapter 96
<a href="#">NashHorn</a>	第96章	<a href="#">Natali</a>	Chapters 205 and 264
<a href="#">娜塔莉</a>	第205章和第264章	<a href="#">Neeraj</a>	Chapter 259
<a href="#">尼拉杰</a>	第259章	<a href="#">Nepster</a>	Chapter 8
<a href="#">Nepster</a>	第8章	<a href="#">nibarius</a>	Chapter 120
<a href="#">nibarius</a>	第120章	<a href="#">Nick</a>	Chapter 1
<a href="#">尼克</a>	第1章	<a href="#">Nick Cardoso</a>	Chapters 38, 41, 43, 95 and 160
<a href="#">尼克·卡多索</a>	第38章、第41章、第43章、第95章和第160章	<a href="#">Nickan B</a>	Chapter 50
<a href="#">尼坎·B</a>	第50章	<a href="#">Nicolai Weitkemper</a>	Chapter 169
<a href="#">尼古拉伊·韦特凯姆珀</a>	第169章	<a href="#">Nicolas Maltais</a>	Chapter 198
<a href="#">尼古拉斯·马尔泰斯</a>	第198章	<a href="#">Nikita Kurtin</a>	Chapter 37
<a href="#">尼基塔·库尔廷</a>	第37章	<a href="#">niknetniko</a>	Chapter 41
<a href="#">niknetniko</a>	第41章	<a href="#">Nilanchala Panigrahy</a>	Chapter 250
<a href="#">尼兰查拉·帕尼格拉希</a>	第250章	<a href="#">Nilesh Singh</a>	Chapter 143
<a href="#">尼莱什·辛格</a>	第143章	<a href="#">Nissim R</a>	Chapter 151
<a href="#">尼西姆·R</a>	第151章	<a href="#">NitZRobotKoder</a>	Chapter 165
<a href="#">NitZRobotKoder</a>	第165章	<a href="#">noob</a>	Chapter 123
<a href="#">新手</a>	第123章	<a href="#">noongiya95</a>	Chapters 37 and 193
<a href="#">noongiya95</a>	第37章和第193章	<a href="#">Nougat Lover</a>	Chapters 14, 45, 97 and 118
<a href="#">牛轧糖爱好者</a>	第14章、第45章、第97章和第118章	<a href="#">null pointer</a>	Chapter 113
<a href="#">空指针</a>	第113章	<a href="#">Oknesif</a>	Chapter 209
<a href="#">奥克内西夫</a>	第209章	<a href="#">Oleksandr</a>	Chapter 219
<a href="#">亚历山大</a>	第219章	<a href="#">Olu</a>	Chapter 57
<a href="#">奥卢</a>	第57章		

奥马尔·阿弗拉克	第229章	<a href="#">Omar Aflak</a>	Chapter 229
奥马尔·阿尔·哈拉比	第96章	<a href="#">Omar Al Halabi</a>	Chapter 96
once2go	第92章	<a href="#">once2go</a>	Chapter 92
奥尼克	第59章	<a href="#">Onik</a>	Chapter 59
奥努尔	第31章	<a href="#">Onur</a>	Chapter 31
orelzion	第73章	<a href="#">orelzion</a>	Chapter 73
奥伦	第258章	<a href="#">Oren</a>	Chapter 258
originx	第252章	<a href="#">originx</a>	Chapter 252
oshurmamadov	第37章和第92章	<a href="#">oshurmamadov</a>	Chapters 37 and 92
巴勃罗·巴克斯特	第16章、第35章、第82章、第98章和第239章	<a href="#">Pablo Baxter</a>	Chapters 16, 35, 82, 98 and 239
潘卡杰·库马尔	第26章	<a href="#">Pankaj Kumar</a>	Chapter 26
帕雷什·马亚尼	第97章	<a href="#">Paresh Mayani</a>	Chapter 97
帕尔萨尼娅·哈迪克	第42章	<a href="#">Parsania Hardik</a>	Chapter 42
帕特里克·达蒂利奥	第8章、第76章和第253章	<a href="#">Patrick Dattilio</a>	Chapters 8, 76 and 253
保罗·拉默茨马	第41章	<a href="#">Paul Lammertsma</a>	Chapter 41
帕维尔·杜罗夫	第56章、第205章和第264章	<a href="#">Pavel Durov</a>	Chapters 56, 205 and 264
帕维尔·斯特列琴科	第47章	<a href="#">Pavel Strelchenko</a>	Chapter 47
帕夫尼特·辛格	第1章、第41章和第47章	<a href="#">Pavneet Singh</a>	Chapters 1, 41 and 47
帕维尔·卡拉	第40章	<a href="#">Pawel Cala</a>	Chapter 40
佩德罗·瓦雷拉	第197章	<a href="#">Pedro Varela</a>	Chapter 197
彼得·泰勒	第120章	<a href="#">Peter Taylor</a>	Chapter 120
范文灵	第2章和第8章	<a href="#">Phan Van Linh</a>	Chapters 2 and 8
菲尔	第72章	<a href="#">Phil</a>	Chapter 72
菲尔实验室	第90章和第262章	<a href="#">PhilLab</a>	Chapters 90 and 262
皮纳基·阿查里亚	第96章	<a href="#">Pinaki Acharya</a>	Chapter 96
piotrek1543	第69章	<a href="#">piotrek1543</a>	Chapter 69
皮尤什	第76章和第83章	<a href="#">Piyush</a>	Chapters 76 and 83
蓬帕特	第88章和第132章	<a href="#">Pongpat</a>	Chapters 88 and 132
普拉杜姆·库马尔·马汉塔	第200章	<a href="#">Pradumn Kumar Mahanta</a>	Chapter 200
普拉卡什·巴拉	第24章	<a href="#">Prakash Bala</a>	Chapter 24
普拉奈	第92章、第125章和第263章	<a href="#">pRaNaY</a>	Chapters 92, 125 and 263
普拉蒂克·布塔尼	第9章、第21章、第26章、第147章、第202章和第245章	<a href="#">Pratik Butani</a>	Chapters 9, 21, 26, 147, 202 and 245
privatestaticint	第157章	<a href="#">privatestaticint</a>	Chapter 157
普里亚·帕拉沙尔	第249章	<a href="#">PRIYA PARASHAR</a>	Chapter 249
普里扬克·帕特尔	第6、26和162章	<a href="#">Priyank Patel</a>	Chapters 6, 26 and 162
专业模式	第1和102章	<a href="#">Pro Mode</a>	Chapters 1 and 102
普朗纳奇	第219章	<a href="#">Prownage</a>	Chapter 219
PSN	第1章	<a href="#">PSN</a>	Chapter 1
R. 扎戈尔斯基	第2、38、47、56、65、73、83、84、88、131、176、205、255和264章	<a href="#">R. Zagórski</a>	Chapters 2, 38, 47, 56, 65, 73, 83, 84, 88, 131, 176, 205, 255 and 264
拉詹·KS	第40章	<a href="#">rajan ks</a>	Chapter 40
拉杰什	第8、37、41、62、72和93章	<a href="#">Rajesh</a>	Chapters 8, 37, 41, 62, 72 and 93
拉克希特·纳瓦尼	第211章	<a href="#">Rakshit Nawani</a>	Chapter 211
拉曼	第250章	<a href="#">Raman</a>	Chapter 250
拉姆齐·哈桑	第61章	<a href="#">Ramzy Hassan</a>	Chapter 61
兰维尔	第247章	<a href="#">Ranveer</a>	Chapter 247
拉苏尔·米里	第76章	<a href="#">Rasoul Miri</a>	Chapter 76
拉维·鲁帕雷利亚	第39章和第62章	<a href="#">Ravi Rupareliya</a>	Chapters 39 and 62
rciovati	第41章和第47章	<a href="#">rciovati</a>	Chapters 41 and 47
里亚兹·穆尔谢德	第16章、第41章、第47章和第56章	<a href="#">Reaz Murshed</a>	Chapters 16, 41, 47 and 56
RediOne1	第13章、第41章、第53章、第62章和第80章	<a href="#">RediOne1</a>	Chapters 13, 41, 53, 62 and 80
雷德曼	第84章	<a href="#">Redman</a>	Chapter 84
reflective_mind	第171章	<a href="#">reflective_mind</a>	Chapter 171
rekire	第8章、第15章、第41章、第44章、第47章、第218章、第255章和第263章	<a href="#">rekire</a>	Chapters 8, 15, 41, 44, 47, 218, 255 and 263

雷万特·戈皮	第47章	<a href="#">Revanth Gopi</a>	Chapter 47
reVerse	第243章	<a href="#">reVerse</a>	Chapter 243
ReverseCold	第241章	<a href="#">ReverseCold</a>	Chapter 241
里卡多·维埃拉	第43章、第131章和第135章	<a href="#">Ricardo Vieira</a>	Chapters 43, 131 and 135
ridsatrio	第37章和第97章	<a href="#">ridsatrio</a>	Chapters 37 and 97
里施	第258章	<a href="#">Risch</a>	Chapter 258
里什布·沙尔马	第264章	<a href="#">RishbhSharma</a>	Chapter 264
罗伯特	第122章	<a href="#">Robert</a>	Chapter 122
罗伯特·班雅伊	第192章	<a href="#">Robert Banyai</a>	Chapter 192
罗伯托·贝坦库尔特	第213章	<a href="#">Roberto Betancourt</a>	Chapter 213
罗宾·迪克霍夫	第227章	<a href="#">Robin Dijkhof</a>	Chapter 227
罗汉·阿罗拉	第99章	<a href="#">Rohan Arora</a>	Chapter 99
罗希特·阿里亚	第40章、第61章、第78章和第123章	<a href="#">Rohit Arya</a>	Chapters 40, 61, 78 and 123
罗尔夫·少	第253章	<a href="#">Rolf 少</a>	Chapter 253
罗穆·迪齐	第181章	<a href="#">Romu Dizzy</a>	Chapter 181
罗萨里奥·佩雷拉·费尔南德斯	第76章和第139章	<a href="#">Rosário Pereira Fernandes</a>	Chapters 76 and 139
鲁宾·内利库纳图	第44章、第98章和第154章	<a href="#">Rubin Nellikunnathu</a>	Chapters 44, 98 and 154
鲁帕利	第69章、第139章和第165章	<a href="#">Rupali</a>	Chapters 69, 139 and 165
russjr08	第41章	<a href="#">russjr08</a>	Chapter 41
russt	第1章和第263章	<a href="#">russt</a>	Chapters 1 and 263
S.D.	第90章	<a href="#">S.D.</a>	Chapter 90
S.R	第14章、第68章和第161章	<a href="#">S.R</a>	Chapters 14, 68 and 161
赛义德	第7章和第229章	<a href="#">Saeed</a>	Chapters 7 and 229
萨加尔·查瓦达	第16章和第37章	<a href="#">Sagar Chavada</a>	Chapters 16 and 37
萨姆·贾德	第61章	<a href="#">Sam Judd</a>	Chapter 61
萨米拉·拉克希塔	第98章	<a href="#">sameera lakshitha</a>	Chapter 98
samgak	第185章	<a href="#">samgak</a>	Chapter 185
Sammy T	第78章	<a href="#">Sammy T</a>	Chapter 78
SANAT	第15章	<a href="#">SANAT</a>	Chapter 15
桑凯特·贝尔德	第16、92和229章	<a href="#">Sanket Berde</a>	Chapters 16, 92 and 229
Sanoop	第37和117章	<a href="#">Sanoop</a>	Chapters 37 and 117
萨桑克·桑卡瓦利	第16章	<a href="#">Sasank Sunkavalli</a>	Chapter 16
Sashabrava	第22章	<a href="#">Sashabrava</a>	Chapter 22
索尔	第1章	<a href="#">saul</a>	Chapter 1
索拉夫	第78章	<a href="#">saurav</a>	Chapter 78
萨文	第167章	<a href="#">Saveen</a>	Chapter 167
塞贡·法米萨	第39章	<a href="#">Segun Famisa</a>	Chapter 39
塞弗尔	第15章	<a href="#">Sevle</a>	Chapter 15
shadygoneinsane	第222章	<a href="#">shadygoneinsane</a>	Chapter 222
shahharshil46	第229章	<a href="#">shahharshil46</a>	Chapter 229
ShahiM	第162章	<a href="#">ShahiM</a>	Chapter 162
沙利尼	第232章	<a href="#">shalini</a>	Chapter 232
尚塔努·保罗	第74章	<a href="#">Shantanu Paul</a>	Chapter 74
沙尚特	第250章	<a href="#">Shashanth</a>	Chapter 250
谢卡尔	第189章	<a href="#">Shekhar</a>	Chapter 189
希卡尔·班萨尔	第230章	<a href="#">shikhar bansal</a>	Chapter 230
希尼尔·M·S	第40章、第92章和第164章	<a href="#">Shinil M S</a>	Chapters 40, 92 and 164
Shirane85	第88章	<a href="#">Shirane85</a>	Chapter 88
ShivBuyya	第40章和第62章	<a href="#">ShivBuyya</a>	Chapters 40 and 62
shtolik	第24章和第117章	<a href="#">shtolik</a>	Chapters 24 and 117
舒布哈姆·舒克拉	第230章	<a href="#">Shubham Shukla</a>	Chapter 230
悉达特·韦努	第1章	<a href="#">Siddharth Venu</a>	Chapter 1
西蒙	第41章和第116章	<a href="#">Simon</a>	Chapters 41 and 116

西蒙·舒伯特  
西蒙娜·卡莱蒂  
Simplans  
SimplyProgrammer  
SC爵士  
Smit.Satodia  
斯内赫·潘迪亚  
索海尔·扎希德  
索鲁什A  
太空飞机  
ssimm  
斯塔诺伊科维奇  
斯特凡·马蒂斯  
史蒂夫.P  
still\_learning  
stkent  
sud007  
苏吉特·尼赖库拉坦  
苏克里特·库马尔  
sukumar  
Sup  
Suragch  
苏雷什·库马尔  
清扫工  
塔格  
tainy  
塔尔哈·米尔  
TameHog  
Tanis.7x  
TDG  
特贾斯·帕瓦尔  
theFunkyEngineer  
THelper  
thetonrifles  
thiagolr  
托马斯·伊索  
托马斯·蒂博  
田  
蒂姆·克拉嫩  
蒂莫·贝尔  
托米克  
Tot Zam  
tpk  
TR4Android  
特里纳德·科亚  
图多尔·卢卡  
tynn  
ubuntudroid  
字词库  
乌里尔·卡里略  
user1506104  
USKMobility  
乌塔姆·潘查萨拉

第118章  
第74章和第262章  
第1章和第41章  
第37章  
第8章和第97章  
第18章  
第4章  
第41章、第129章和第147章  
第41章  
第173章  
第118章  
第41章  
第33章  
第59章  
第29章和第264章  
第14章、第15章、第25章、第26章、第37章和第153章  
第2章、第4章、第28章、第118章、第140章和第165章  
第177章和第242章  
第61、183、205、239和264章  
第21章  
第28章  
第67章、第183章和第239章  
第71章  
第59章  
第41章  
第8章  
第126章  
第38章、第39章、第124章和第247章  
第116章和第263章  
第250章  
第41章和第264章  
第262章  
第16章  
第59章、第263章和第264章  
第64章  
第264章  
第2、8、38、41、44、47、218和262章  
第264章  
第8和219章  
第254章  
第151章  
第16章  
第98章  
第28、69和70章  
第146章  
第30章  
第106、109、114、211和214章  
第8章  
第92章  
第81章和第227章  
第250章  
第265章  
第50章和第87章

Simon Schubert  
Simone Carletti  
Simplans  
SimplyProgrammer  
Sir SC  
Smit.Satodia  
Sneh Pandya  
Sohail Zahid  
SoroushA  
spaceplane  
ssimm  
Stanojkovic  
Stephane Mathis  
Steve.P  
still\_learning  
stkent  
sud007  
Sujith Niraikulathan  
Sukrit Kumar  
sukumar  
Sup  
Suragch  
Suresh Kumar  
Sweeper  
Täg  
tainy  
Talha Mir  
TameHog  
Tanis.7x  
TDG  
Tejas Pawar  
theFunkyEngineer  
THelper  
thetonrifles  
thiagolr  
Thomas Easo  
ThomasThiebaud  
Tien  
Tim Kranen  
Timo Bähr  
Tomik  
Tot Zam  
tpk  
TR4Android  
TRINADH KOYA  
Tudor Luca  
tynn  
ubuntudroid  
Ufkoku  
Uriel Carrillo  
user1506104  
USKMobility  
Uttam Panchasara

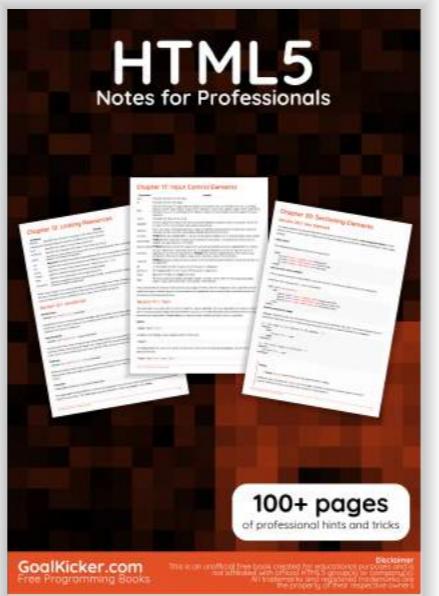
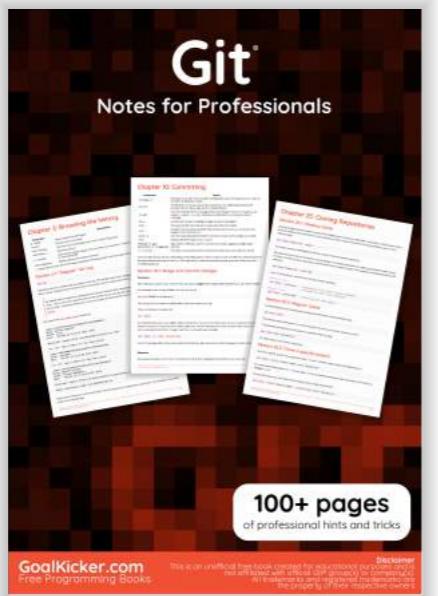
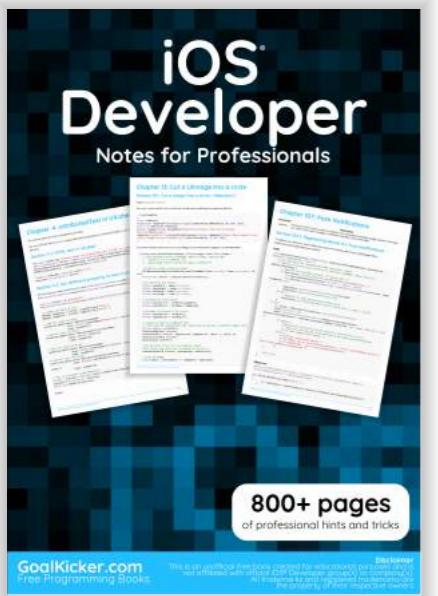
Chapter 118  
Chapters 74 and 262  
Chapters 1 and 41  
Chapter 37  
Chapters 8 and 97  
Chapter 18  
Chapters 1, 9, 14, 16, 37, 61, 96 and 257  
Chapter 4  
Chapters 41, 129 and 147  
Chapter 41  
Chapter 173  
Chapter 118  
Chapter 41  
Chapter 33  
Chapter 59  
Chapters 29 and 264  
Chapters 14, 15, 25, 26, 37 and 153  
Chapters 2, 4, 28, 118, 140 and 165  
Chapters 177 and 242  
Chapters 61, 183, 205, 239 and 264  
Chapter 21  
Chapter 28  
Chapters 67, 183 and 239  
Chapter 71  
Chapter 59  
Chapter 41  
Chapter 8  
Chapter 126  
Chapters 38, 39, 124 and 247  
Chapters 116 and 263  
Chapter 250  
Chapters 41 and 264  
Chapter 262  
Chapter 16  
Chapters 59, 263 and 264  
Chapter 64  
Chapters 2, 8, 38, 41, 44, 47, 218 and 262  
Chapter 264  
Chapters 8 and 219  
Chapter 254  
Chapter 151  
Chapter 16  
Chapter 98  
Chapters 28, 69 and 70  
Chapter 146  
Chapter 30  
Chapters 106, 109, 114, 211 and 214  
Chapter 8  
Chapter 92  
Chapters 81 and 227  
Chapter 250  
Chapter 265  
Chapters 50 and 87

V  
维克多·阿尔贝托斯  
V. 卡柳日纽  
瓦西里·卡布诺夫  
维基  
维尼修斯·巴罗斯  
维奈  
文森特·D.  
vipsy  
维沙尔·普里  
维什瓦纳特·N·P  
维什韦什·贾因库尼娅  
维韦克·米什拉  
弗隆贾特·加希  
弗洛迪米尔·布贝连科  
vrbsm  
武科  
WOrmH0le  
W3hri  
沃伦·费斯 (WarrenFaith)  
webo80  
韦斯顿 (weston)  
威利·查尔默斯三世  
哈维尔·卡佩勒  
哈维尔  
xDragonZ  
y.feizi  
亚沙斯维·马哈希  
亚辛·卡奇马兹  
亚西  
yennsarах  
约格什·乌梅什·瓦伊蒂  
Yojimbo  
younes zeboudj  
尤沙·阿莱尤布  
yuku  
尤里·费多罗夫  
伊薇特·科隆布  
扎卡里·大卫·桑德斯  
泽山·沙比尔  
泽恩特里诺  
ZeroOne  
Zilk  
Zoe  
Héb è  
R N

第56章  
第251章  
第16章  
第227章  
第78章  
第45章  
第41章  
第219章  
第40章和第250章  
第36章  
第98章  
第121章  
第1、5、38和41章  
第39、47和83章  
第40和97章  
第92章  
第78章和第123章  
第72章和第133章  
第128章  
第14章  
第43章和第54章  
第69章  
第12章、第37章和第238章  
第37、142、156、218、227和264章  
第38章  
第223章  
第135章  
第146章  
第17和53章  
第47章和第264章  
第39章和第231章  
第161章  
第62章  
第41章、第218章和第264章  
第154章  
第47章  
第1、8、9、14、28、40、41、47、205、218、219、227、263和264章  
第57和104章  
第263章  
第34和235章  
第110章  
第13和61章  
第16章和第250章  
第1章、第3章、第62章、第94章、第105章、第145章、第148章、第208章和第242章  
第4章和第43章  
第1章和第250章

V  
Víctor Albertos  
V. Kalyuzhnyu  
Vasily Kabunov  
Vicky  
Vinícius Barros  
Vinay  
Vincent D.  
vipsy  
Vishal Puri  
VISHWANATH N P  
Vishwesh Jainkuniya  
Vivek Mishra  
Vlonjat Gashi  
Volodymyr Buberenko  
vrbsm  
Vucko  
WOrmH0le  
W3hri  
WarrenFaith  
webo80  
weston  
Willie Chalmers III  
Xaver Kapeller  
Xavier  
xDragonZ  
y.feizi  
Yashaswi Maharshi  
Yasin Kaçmaz  
Yassie  
yennsarах  
Yogesh Umesh Vaity  
Yojimbo  
younes zeboudj  
Yousha Aleayoub  
yuku  
Yury Fedorov  
Yvette Colomb  
Zachary David Saunders  
Zeeshan Shabbir  
Zerntrino  
ZeroOne  
Zilk  
Zoe  
Héb è  
R N

## 你可能还喜欢



## You may also like

