Android移动应用开发基础教程

讲授：葛新

第3章 UI设计

本章主要内容：

- 布局
- 通用UI组件
- 消息通知
- 对话框
- 菜单
- ListView
- RecyclerView

3.1 布局

布局是Android应用程序的界面定义。布局中的所有界面元素，都是视图（View）或视图组（ViewGroup）对象。一个布局首先是一个视图组对象，然后在视图组对象中添加子视图组对象或者视图对象。

本节主要内容：

1. 视图和视图组
2. 布局的定义方法
3. 线性布局LinerLayout
4. 相对布局RelativeLayout
5. 帧布局FrameLayout

3.1.1 视图和视图组

- 视图对象用于在屏幕上绘制可与用户交互的界面元素。一个视图占据一块矩形屏幕区域，并通过属性设置来渲染此区域。视图区域也可设置是否可见、是否可获得焦点，也可处理区域中发生的事件（用户触摸、拖动等等）。
- 在Android中，View类是所有用于设计界面组成元素的基类，Button、CheckBox、ExitView、ImageView、ProgressBar、TextView以及其他的UI组件，都是View类的子类或子类的派生类。
- 视图组是一种特殊的视图，它不具有可见性，而是一种容器。在视图组中可包含视图组和视图。ViewGroup类是View类的一个子类，它又是各种布局类的基类。常用的布局类有LinearLayout（线性布局）、RelativeLayout（相对布局）和FrameLayout（帧布局）类等。

3.1.2 布局的定义方法

- 可通过两种方法来定义布局：XML定义和代码定义
- 布局的XML定义是使用Android的XML词汇，以文本的方式在快速设计UI布局及其包含的界面元素。

```
<?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:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="这是自定义的另一个布局" />
    <Button
        android:text="设置"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnSet"/>
</LinearLayout>
```

3.1.3 线性布局LinerLayout

- LinerLayout是一个视图组，它按照垂直或水平方式按顺序排列内部的视图或视图组对象。线性布局中，每行或每列中只允许有一个子视图。
  - android:gravity：设置内部组件的显示位置。
  - android:orientation：设置内部组件的排列方向，常量horizontal表示水平排列，vertical（默认值）表示垂直排列。
  - android:background：设置一个drawable资源作为背景。
  - android:id：设置布局ID.
  - android:padding：设置所有边距的统一值
  - android:paddingBottom：设置底部边距
  - android:paddingLeft：设置左边距
  - android:paddingRight：设置右边距
  - android:paddingTop：设置顶部边距

```
<?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">
    <Button   android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="按钮1" />
    <Button   android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="按钮2" />
    <Button   android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="按钮3" />
</LinearLayout>
```

3.1.4 相对布局RelativeLayout

- RelativeLayout是一个视图组，它按照相对位置来排列各个子视图。
- 在使用相对布局时，子视图默认位于左上角，可使用下列属性来控制子视图的位置：
  - android:layout_alignParentTop：设置为true时，子视图的上边框与父视图的上边框对齐。
  - android:layout_centerVertical：设置为true时，子视图在垂直方向上的位于父视图中间位置。
  - android:layout_centerHorizontal：设置为true时，子视图在水平方向上的位于父视图中间位置。
  - android:layout_below：设置一个控件ID，子视图位于该控件下方。
  - android:layout_toRightOf：设置一个控件ID，子视图位于该控件右侧。
  - android:layout_toLeftOf：设置一个控件ID，子视图位于该控件左侧。

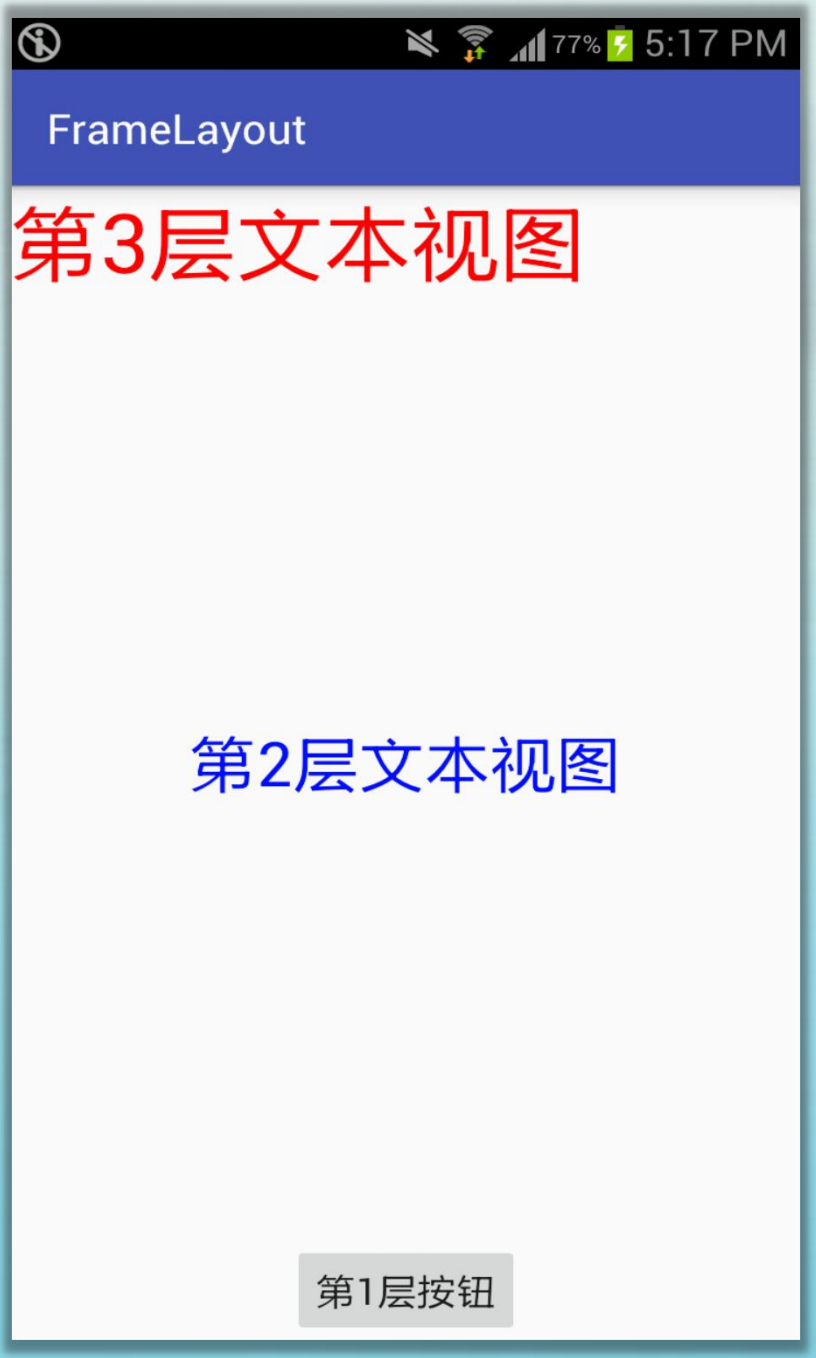
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"   android:id="@+id/activity_main"
    android:layout_width="match_parent"       android:layout_height="match_parent"
    tools:context="com.example.xbg.relativelayout.MainActivity">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:hint="输入用户名"
        android:id="@+id/editText1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editText1"
        android:hint="输入密码"
        android:id="@+id/editText2" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/editText2"
        android:layout_alignParentRight="true"
        android:text="确定" />
</RelativeLayout>
```

3.1.5 帧布局FrameLayout

- 帧布局是一种特殊的布局，它以层叠的方式显示布局中的多个控件，最后添加的控件位于最前面。
- 默认情况下，控件位于帧布局的左上角。可通过控件的android:layout_gravity属性控制其位置。android:layout_gravity属性可设置为下列值：
  - top：控件位于布局顶部。
  - bottom：控件位于布局底部。单独使用时等价于“left|bottom”。
  - left：控件位于布局左侧。
  - right：控件位于布局右侧。单独使用时等价于“top|right”。
  - center：控件位于布局中心。
  - center_vertical：控件位于垂直方向上的中间位置。单独使用时等价于“left|center_vertical”。
  - center_horizontal：控件位于水平方向上的中间位置。单独使用时等价于“top|center_ horizontal”。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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"
    tools:context="com.example.xbg.framelayout.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="40dip"
        android:textColor="#ff0000"
        android:text="第3层文本视图"
        android:id="@+id/textView1" />
    <TextView
        android:text="第2层文本视图"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000dff"
        android:textSize="30dip"
        android:id="@+id/textView2" />
    <Button
        android:text="第1层按钮"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button" />
</FrameLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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"
    tools:context="com.example.xbg.framelayout.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="40dip"
        android:textColor="#ff0000"
        android:text="第3层文本视图"
        android:id="@+id/textView1" />
    <TextView
        android:text="第2层文本视图"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000dff"
        android:textSize="30dip"
        android:layout_gravity="center
        android:id="@+id/textView2" />
    <Button
        android:text="第1层按钮"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center|bottom"
        android:id="@+id/button" />
</FrameLayout>
```



3.2 通用UI组件

本节主要内容：

1. 文本视图（TextView）
2. 按钮（Button）
3. 文本字段（EditText、AutoCompleteTextView）
4. 复选框（CheckBox）
5. 单选按钮（RadioButton）
6. 切换按钮（ToggleButton）
7. 下拉列表（Spinner）
8. 日期选取器（DatePicker）
9. 时间选取器（TimePicker）
10. 拖动条（SeekBar）

3.2.1 文本视图（TextView）

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="hello，极客学院" />
```

- 可使用下列属性设置文本显示效果：
  - android:typeface：设置字体。Android默认支持4中内置字体：normal、sans、serif和monospace。
  - android:textSize：设置字号。
  - android:textColor：设置颜色。
  - android:textStyle：设置文本样式，可设置为bold、italic或bolditalic。

3.2.2 按钮（Button、ImageButton）

```
<Button
        android:text="Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button1" />
<Button
        android:text="Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableLeft="@mipmap/ic_launcher"
        android:id="@+id/button2" />
```

```
<ImageButton
        android:layout_width="wrap_conten
        android:layout_height="wrap_conte
        android:src="@mipmap/ic_launcher"
        android:id="@+id/imageButton1" />
```

- 通常，需要为按钮添加Click事件监听器。一种方法是在android:onClick属性中设置Click事件监听器。例如：

```
<Button
        ……
        android:id="@+id/button1"
        android:onClick="ClickButton1"/>
```

- 其中的ClickButton1是在代码中定义的一个方法，且方法必须是public和void类型。例如：

```
public void ClickButton1(View view){
        TextView tv1= (TextView) findViewById(R.id.textView);
        tv1.setText("单击按钮Button1");
}
```

- 另一种为按钮添加Click事件监听器的方法是在代码执行setOnClickListener()方法。例如：

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button bt2=(Button) findViewById(R.id.button2);
        bt2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TextView tv1= (TextView) findViewById(R.id.textView);
                tv1.setText("单击按钮Button2");
            }
        });
    }
```

3.2.3 文本字段（EditText、AutoCompleteTextView）

- 文本字段控件用于接收用户输入，可使用android:inputType属性定义各种输入行为准则。常用android:inputType属性值如下：
  - text：允许输入各种文本。
  - textMultiLine：允许输入多行文本。
  - textEmailAddress：只允许输入Email地址。
  - textPassword：用于输入密码。
  - number：只允许输入数字。
  - phone：用于输入电话号码。
  - datetime：用于输入日期时间。

EditText

```
<EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:id="@+id/editText" />
```

AutoCompleteTextView

- 第1步：在布局文件中添加AutoCompleteTextView控件。例如：

```
<AutoCompleteTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="1"
        android:id="@+id/autoCompleteTextView"  />
```

- 第2步：在资源文件res/values/strings.xml中定义提供自动完成提示的字符串数组资源。例如：

```
<resources>
……
    <string-array name="select_array">
        <item>cable</item>
        <item>china</item>
        <item>Chinese</item>
        <item>Check</item>
    </string-array>
</resources>
```

- 第3步：为AutoCompleteTextView绑定提供自动完成提示的适配器。例如：

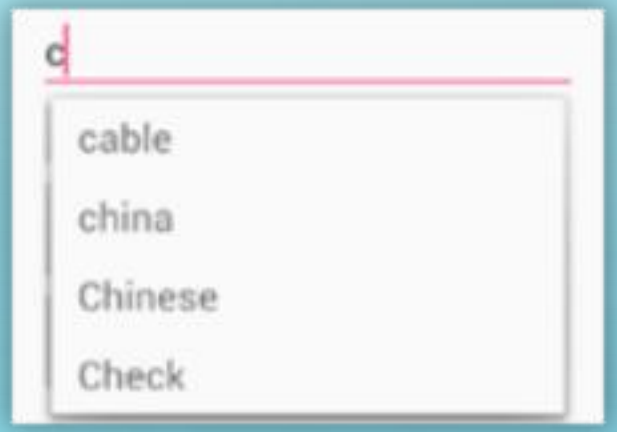
```
AutoCompleteTextView
act=(AutoCompleteTextView)findViewById(R.id.autoCompleteTe
xtView);

String[] selects =
getResources().getStringArray(R.array.select_array);

ArrayAdapter<String> adapter =

new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, selects);

act.setAdapter(adapter);
```



3.2.4 复选框（CheckBox

```
<CheckBox
    android:text="加粗"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/checkBox1" android:onClick="ClickCheckBox1" />
  <CheckBox
    android:text="倾斜"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/checkBox2" android:onClick="ClickCheckBox2" />
```

android:onClick属性为复选框绑定了Click事件监听器，处理复选框Click事件。例如，下面的代码实现在单击复选框时，改变文本视图的样式：

```
private boolean checked1;
public void ClickCheckBox1(View view){
        checked1 = ((CheckBox) view).isChecked();
        ChangeTextViewStyle();
    }
    private boolean checked2;
    public void ClickCheckBox2(View view){
        checked2 = ((CheckBox) view).isChecked();
        ChangeTextViewStyle();
    }
```

```
ChangeTextViewStyle(){
        TextView tv1= (TextView) findViewById(R.id.textView);
        Typeface tf= tv1.getTypeface();
        int style=0;
        if(checked1){
            style=1;
            if(checked2){style=3;}
        }
        else
            if(checked2){style=2;}
        tv1.setTypeface(tf,style);
}
```

3.2.5 单选按钮（RadioButton）

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:checkedButton="@+id/radioButton1" >
    <RadioButton
      android:text="蓝色"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/radioButton1"
      android:layout_weight="1" android:onClick="ClickRadio" />
```

```
<RadioButton
        android:text="红色"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/radioButton2"
        android:layout_weight="1" android:onClick="ClickRadio"/>
    <RadioButton
        android:text="绿色"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/radioButton3"
        android:layout_weight="1" android:onClick="ClickRadio"/>
</RadioGroup>
```

- android:onClick属性为单选按钮绑定Click事件监听器。例如在颜色颜色时更改TextView控件颜色：

```
public void  ClickRadio(View view){
        TextView tv1= (TextView) findViewById(R.id.textView);
        switch(view.getId()) {
            case R.id.radioButton1:
                tv1.setTextColor(Color.rgb(0,0,255));
                break;
            case R.id.radioButton2:
                tv1.setTextColor(Color.rgb(255,0,0));
                break;
            case R.id.radioButton3:
                tv1.setTextColor(Color.rgb(0,255,0));
        }
    }
```

3.2.6 切换按钮（ToggleButton）

```
<ToggleButton
    android:textOff="显示背景图片"  android:textOn="隐藏背景图片"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/toggleButton"  />
```

- 在代码中可调用setOnCheckedChangeListener()方法为切换按钮绑定事件监听器，处理其状态变化。例如，下面的代码利用切换按钮设置或清除布局背景：

```
ToggleButton toggle = (ToggleButton) findViewById(R.id.toggleButton);
        toggle.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                LinearLayout layout=(LinearLayout)findViewById(R.id.activity_main);
                if (isChecked) {
                    layout.setBackgroundResource(R.drawable.back); //为布局设置背景图片
                } else {
                    layout.setBackgroundResource(0);//清除布局背景
                }
            }
        });
```

3.2.7 微调框（Spinner）

- 微调框也可称为下拉列表，它提供一组预定选项供用户选择。默认情况下，微调框只显示当前选中项。微调框与AutoCompleteTextView类似，但微调框不提供输入功能，只在被触摸时展开下拉列表。
- 下面的代码为布局添加一个微调框：（本节实例项目：源代码\03\LearnUIComponent2）

```
<Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/spinner" />
```

3.2.8 图片视图（ImageView）

- 图片视图控件用于在界面中显示图片。使用图片视图控件时，需提供准备图片，并将图片放在资源文件夹drawable中。
- 例如，下面的代码为布局添加一个图片视图控件：（本节实例项目：源代码\03\ LearnUIComponent2）

```
<ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/run"
        android:id="@+id/imageView" />
```

3.2.9 进度条（ProgressBar）

- 进度条控件通常用于表示程序正在后台处理数据，避免用户枯燥的等待。例如，下面的代码为布局添加一个进度条：（本节实例项目：源代码\03\LearnUIComponent2）

```
<ProgressBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/progressBar" />
```

- 进度条有四种样式：
  - 大图标（progressBarStyleLarge）
  - 中等图标（默认样式，progressBarStyle）
  - 小图标（progressBarStyleSmall）
  - 水平条（progressBarStyleHorizontal）

3.2.10 拖动条（SeekBar

- 拖动条可拖动滑块的位置来获得标识的数值。例如，下面的代码为布局添加一个拖动条，并将拖动条最大值设置为100：（本节实例项目：源代码\03\LearnUIComponent2）

```
<SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:id="@+id/seekBar" />
```

3.3 消息通知

在Android应用中，可采用Toast或Notification两种方式向用户提供消息通知。

本节主要内容：

1. 使用Toast
2. 使用Notification

3.3.1 使用Toast

- Toast是在应用运行期间，通过类似于对话框的方式向用户显示消息提示。Toast只占用很少的屏幕，并会在一段时间后自动消失。
- 例如，下面的代码来创建并显示Toast通知：

```
Context context=getApplicationContext();                    //获得应用上下文
String text="这是一个较长时间的Toast";                        //准备Taost中显示的文本
int duration=Toast.LENGTH_LONG;                             //用于设置Toast显示时间
Toast toast=Toast.makeText(context,text,duration);          //生成Toast对象
toast.show();                                               //显示Toast通知
```

3.3.2 使用Notification

- Notification通知首先在通知区域（也称状态栏。）中显示通知图标，用户展开抽屉式通知栏时，查看通知的详细信息。图3-19显示了一个通知区域

- 图3-20显示了抽屉式通知栏。一个通知通常由图标、标题和内容等组成。

- 创建一个简单的通知通常包含下列步骤。
- 第一步：创建NotificationCompat.Builder对象。例如：（本节实例项目：源代码\03\LearnNotification）

```
NotificationCompat.Builder builder=
                        new NotificationCompat.Builder(MainActivity.this);
```

- 第二步：调用NotificationCompat.Builder对象方法设置通知相关内容。例如：

```
builder.setSmallIcon(R.drawable.smallico);                    //设置通知小图标
builder.setContentTitle("嗨，你有一个新消息！");              //设置通知标题
builder.setContentText("你已经学会了创建Notification了。");//设置通知内容
builder.setAutoCancel(true);                    //设置自动删除通知
```

- 第三步：创建在抽屉式通知栏中单击通知时启动活动的Intent（如果仅仅需要显示通知内容，不提供通知单击响应，则该步骤可以省略。）。例如：

```
Intent resultIntent=
                    new Intent(MainActivity.this,NotificationActivity.class);
TaskStackBuilder stackBuilder=TaskStackBuilder.create(MainActivity.this);
stackBuilder.addParentStack(NotificationActivity.class);
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent=
stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(resultPendingIntent);
```

- 第四步：创建Notification对象。例如：

```
Notification notification=builder.build();
```

- 第五步：创建NotificationManager对象显示通知。例如：

```
NotificationManager manager=
(NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
manager.notify(NOTIFICATION_ID,notification);
```

3.4 对话框

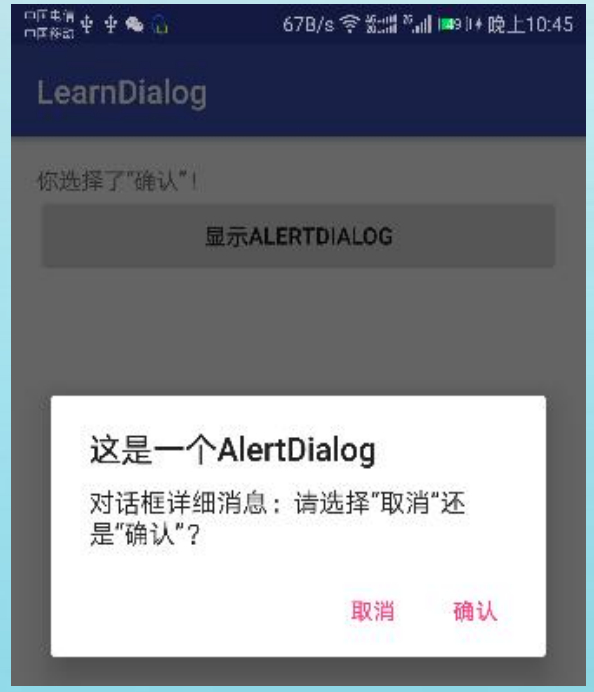
对话框用于在程序运行期间显示一些重要信息，或者用对话框与用户交互。对话框置顶于界面最前面，屏蔽其他所有控件的交互能力。

本节主要内容：

1. AlertDialog
2. ProgressDialog
3. DatePickerDialog
4. TimePickerDialog

3.4.1 AlertDialog

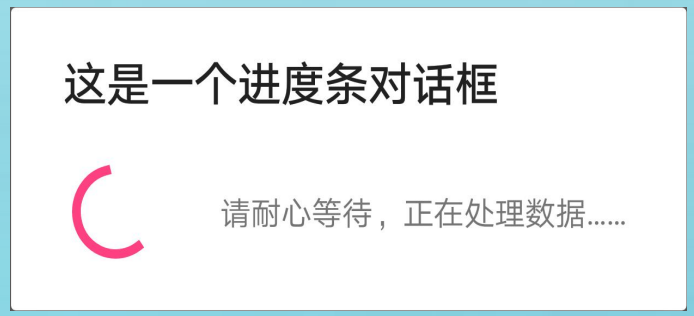
- AlertDialog在对话框中显示警告提示信息，用户可在对话框中选择取消或确认操作。例如，下面的代码用于显示AlertDialog：（本节实例项目：源代码\03\LearnDialog）



3.4.2 ProgressDialog

- ProgressDialog与AlertDialog类似，都可弹出一个对话框，并置顶屏蔽其他控件的交互能力。ProgressDialog可在对话框中显示一个进度条。
- 例如，下面的代码用于显示ProgressDialog。（本节实例项目：源代码\03\LearnDialog）

```
public void  showProgressDialog(View view){
        ProgressDialog progressDialog=new ProgressDialog(MainActivity.this);//创建对话框
        progressDialog.setTitle("这是一个进度条对话框");//设置标题
        progressDialog.setMessage("请耐心等待，正在处理数据……");//设置消息
        progressDialog.setCancelable(true);//设置可取消
        progressDialog.show();//显示对话框
    }
```

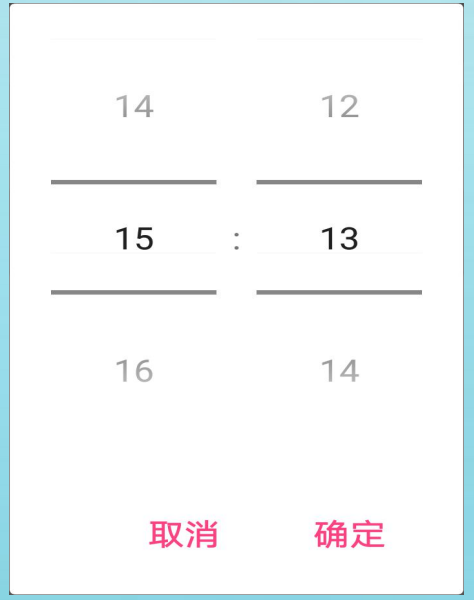


3.4.3 DatePickerDialog

- DatePickerDialog用于显示日期选取对话框，定义OnDateSetListener监听器，实现onDateSet()方法可获得在对话框中选取的日期。
- 例如，下面的代码使用DatePickerDialog显示日期选取对话框，并将选取的日期显示在TextView中。（本节实例项目：源代码\03\LearnDialog）

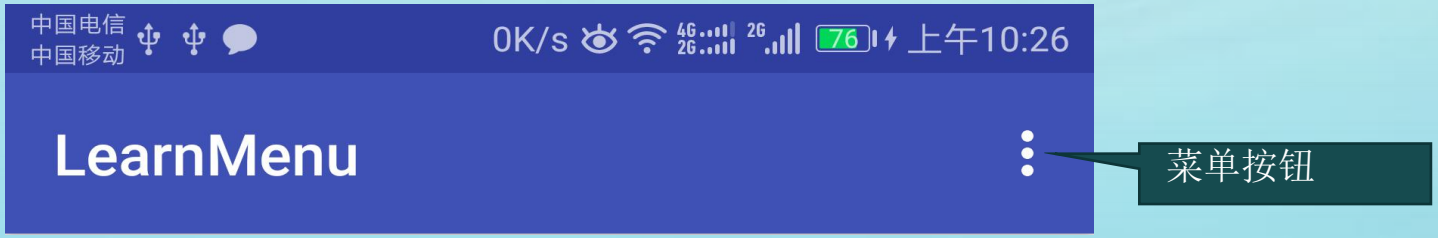
3.4.4 TimePickerDialog

- TimePickerDialog用于显示时间选取对话框，定义OnTimeSetListener监听器，实现onTimeSet()方法可获得在对话框中选取的时间。
- 例如，下面的代码使用TimePickerDialog显示时间选取对话框，并将选取的时间显示在TextView中。（本节实例项目：源代码\03\LearnDialog）



3.5 菜单

- 在手机和平板等设备中，因为空间有限，不再像PC一样为应用程序配置菜单栏。Android提供了一种隐藏的菜单，只在需要的时候展示出来。在图3-25所示的活动标题栏右侧，显示了一个三点符号，这就是菜单按钮。点击菜单按钮可展开活动的菜单。



3.6 ListView

ListView控件用于创建列表，比如联系人、QQ聊天记录等都可用列表来实现。ListView允许用户通过上下滑动的方式将屏幕之外的内容滚动到屏幕内。

本节主要内容：

1. ListView简单用法
2. 自定义ListView列表项布局
3. 处理ListView单击事件

3.6.1 ListView简单用法

- 为布局添加ListView控件非常简单。例如：（本节实例项目：源代码\03\LearnListView）

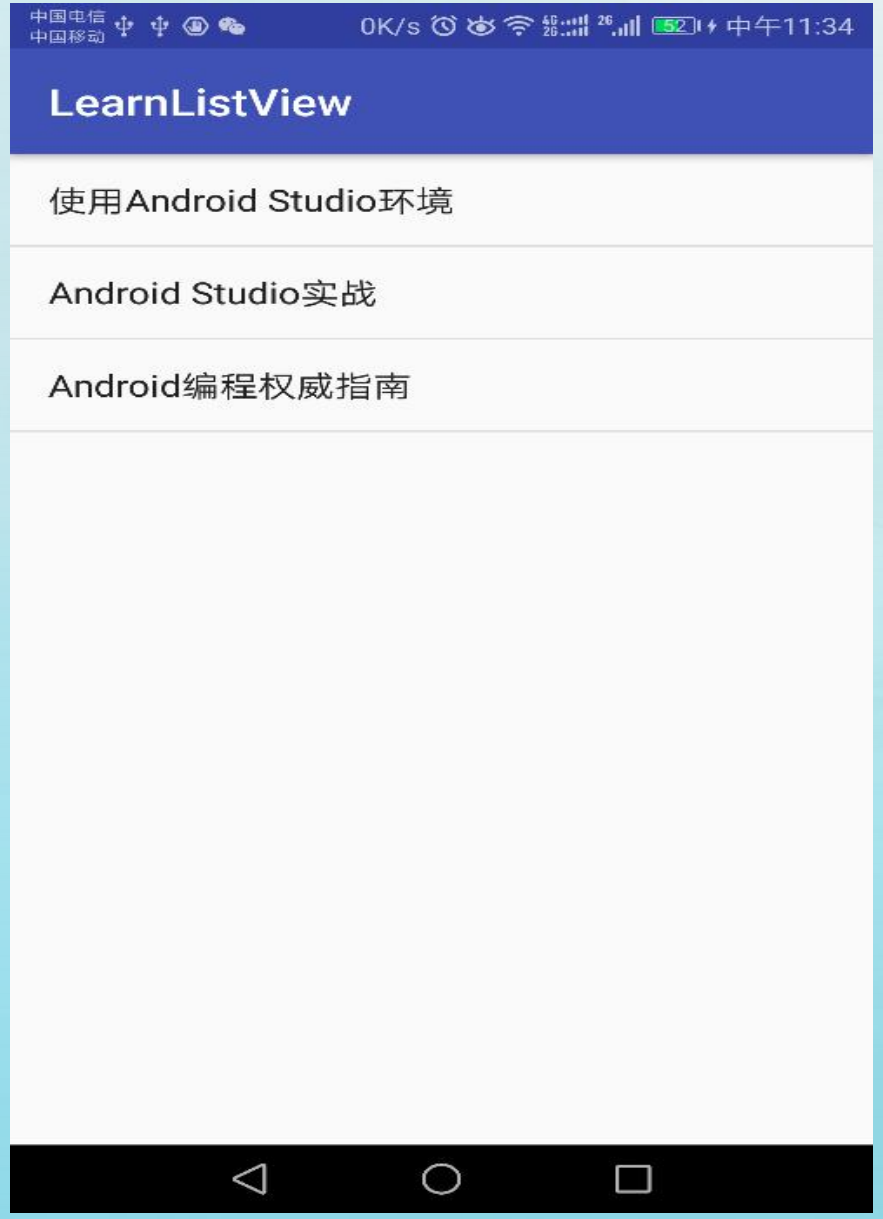
```
<ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/listView"/>
```

- 首先在字符串资源文件string.xml中定义字符串数组。例如：

```
<resources>
    ……
    <string-array name="LearnListViewData">
        <item>使用Android Studio环境</item>
        <item>Android Studio实战</item>
        <item>Android编程权威指南</item>
    </string-array>
</resources>
```

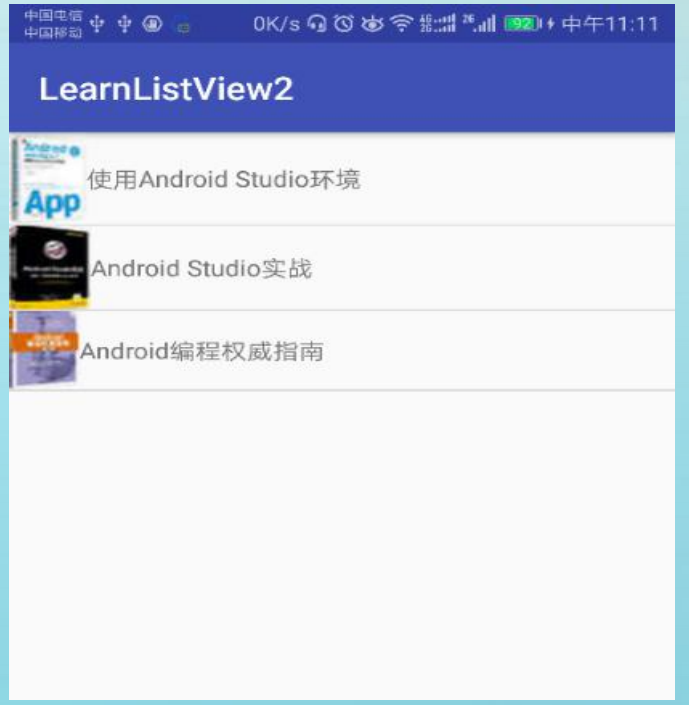
• 然后在代码中用该数组创建ArrayAdapter，并绑定到ListView。例如：

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        String[] data = getResources().getStringArray(R.array.LearnListViewData);
        ArrayAdapter<String> adapter =
                new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
data);
        ListView listView=(ListView)findViewById(R.id.listView);
        listView.setAdapter(adapter);
}
```


中国电信
中国移动
0K/s
中午11:34
LearnListView
使用Android Studio环境
Android Studio实战
Android编程权威指南

3.6.2 自定义ListView列表项布局

- 普通ListView每个列表项只显示一段文本，通过自定义，可以让列表项显示更丰富的内容。
- 下面的实例通过自定义，在列表项中显示图书封面图片和书名。具体操作步骤如下：（本节实例项目：源代码\03\LearnListView2）



3.6.3 处理ListView单击事件

- 要使ListView相应用户单击事件，需要调用setOnItemClickListener()方法绑定OnItemClickListener监听器。例如，修改上节实例中的MainActivity.java，为ListView绑定监听器。代码如下：

```
protected void onCreate(Bundle savedInstanceState) {
……
        ListView listView=(ListView)findViewById(R.id.listView);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Book book=bookList.get(position);
                Toast.makeText(MainActivity.this,book.getName(),Toast.LENGTH_LONG).show();
            }
        });
    }
```

3.7 RecyclerView

ListView在过去的Android应用中发挥了巨大的作用，其功能强大，但缺点明显。ListView只能实现数据垂直滚动，而且在不采取措施优化时性能极差。Android提供了另一个功能更强、效率更高的滚动控件——RecyclerView。RecyclerView可看作是升级版的ListView，Android官方推荐使用RecyclerView来实现滚动列表。

本节主要内容：

1. RecyclerView基本用法
2. 自定义RecyclerView列表项布局
3. RecyclerView布局
4. 处理RecyclerView单击事件

3.7.1 RecyclerView基本用法

- 要使用RecyclerView控件，首先需要在app\buil.gradle文件的dependencies闭包中添加支持库。例如：

```
dependencies {
……
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

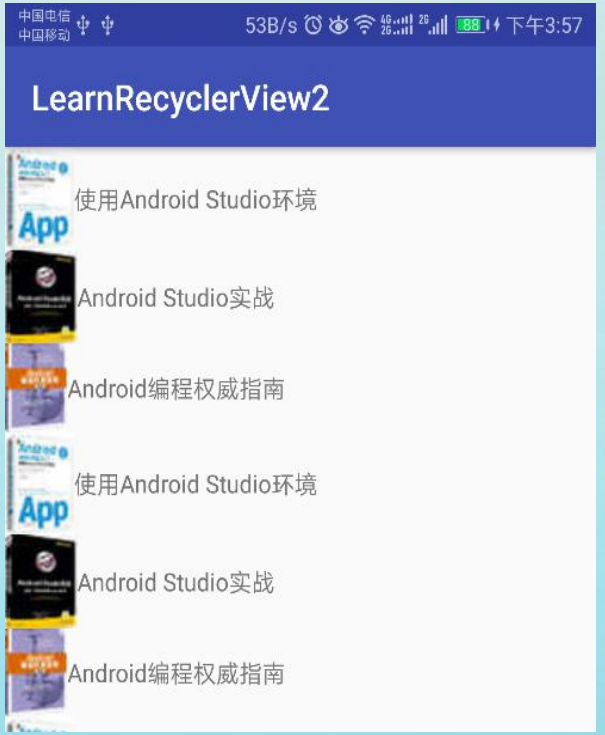
- 在布局文件中，可用代码的代码添加RecyclerView控件：

```
<android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/recyclerView"/>
```

- 下面的实例说明如何使用RecyclerView控件，具体操作步骤如下：（本节实例项目：源代码\03\LearnRecyclerView）

3.7.2 自定义RecyclerView列表项布局

- 与ListView类似，可以使用自定义RecyclerView的列表项布局。
- 下面的实例通过自定义，在RecyclerView列表项中显示图书封面图片和书名。具体操作步骤如下：（本节实例项目：源代码\03\LearnRecyclerView2）



3.7.3 RecyclerView布局

- RecyclerView可以通过setLayoutManager()方法设置布局类型。RecyclerView支持LinearLayoutManager（线性布局）、StaggeredGridLayoutManager（瀑布流布局）和GridLayoutManager（网格布局）等布局。

3.7.4 处理RecyclerView单击事件

- 与ListView不同，RecyclerView没有提供类似的setOnItemClickListener()方法来设置列表项监听器。但可通过RecyclerView列表项中的各个控件来设置监听器。
- 修改3.7.2的LearnRecyclerView2实例中的BookAdapter.java，为列表项控件设置监听器，如下所示：