目录

[一、 android sutdio的安装 4](#_Toc20293)

[1、下载安装（win64下安装3.1.2版本为例） 4](#_Toc8271)

[2、安装Android SDK(同时得先有JDK) 8](#_Toc10258)

[3、环境配置 11](#_Toc30995)

[二、 整体介绍 12](#_Toc607)

[1. 架构:基本附件组件 12](#_Toc4612)

[2. 资源Resources 12](#_Toc16172)

[3. 活动Activity 13](#_Toc18267)

[4. 服务Service 15](#_Toc21513)

[5. AndroidManifest.xml详解 21](#_Toc447)

[6. build.gradle详解 22](#_Toc29450)

[三、 开发 27](#_Toc3799)

[1. 赋值取值 27](#_Toc3986)

[2. 点击事件 27](#_Toc26475)

[3. 主题及颜色 28](#_Toc3705)

[4. Res、图片资源、menu 28](#_Toc6000)

[5. 选中和未选中表现为不同图片不同颜色 29](#_Toc9299)

[6. menu的结构内容 29](#_Toc24514)

[7. 设置图片路径 30](#_Toc15067)

[8. 多语言版本 30](#_Toc9812)

[9. json的处理 30](#_Toc16080)

[10. 渐变遮罩 32](#_Toc12563)

[四、 基础组件及布局 33](#_Toc468)

[1. 重绘标题栏ToolBar 33](#_Toc18657)

[1. 样式隐藏，代码绑定 33](#_Toc30720)

[2. 代码隐藏，前端重绘 35](#_Toc3395)

[2. 组件库AndroidX 35](#_Toc5958)

[3. 弹框Toast/Dialog/Snackbar 36](#_Toc18116)

[4. 滚动插件ScrollView/HorizontalScrollV 38](#_Toc3816)

[5. 浮动按钮FloatingActionButton 38](#_Toc13587)

[6. 重绘标题栏目 39](#_Toc32640)

[7. 引入公共部分 39](#_Toc32610)

[8. 引用网页 WebView 40](#_Toc24687)

[9. touch 40](#_Toc24736)

[10. 下拉标题栏变化AppbarLayou 41](#_Toc26034)

[11. 生成右上脚菜单optionsMenu 44](#_Toc13943)

[五、 intent意图 45](#_Toc4461)

[1.跳转内部页面（显式） 45](#_Toc4400)

[2.跳转内部页面（隐式） 45](#_Toc13988)

[3.跳转网址并选择用什么浏览器打开 45](#_Toc5116)

[4.打电话 45](#_Toc7941)

[5.向下一个活动传参 46](#_Toc12331)

[6.返回数据给上一个活动 46](#_Toc21416)

[六、 Fragment片段 47](#_Toc23375)

[1.创建基本的Fragment 48](#_Toc18741)

[2.Activity点击事件修改Fragment里的内容 49](#_Toc8040)

[3. 动态修改FrameLayout里的fragment页面 50](#_Toc29134)

[4. Fragment的一些技巧 50](#_Toc7566)

[七、 ViewPager 51](#_Toc2987)

[八、 线程及HTTP请求 52](#_Toc7850)

[1.线程 52](#_Toc31062)

[九、 列表：ListView/GridView/RecyclerView 53](#_Toc27459)

[1. ArrayAdapter 只可以传一个文本 53](#_Toc6362)

[2. SimpleAdapter多个String 54](#_Toc10859)

[3. baseAdapter (常用) 56](#_Toc31083)

[4. ListView的点击事件 59](#_Toc18759)

[5. GridView 59](#_Toc2894)

[6. 更新GridView 适配器实例 59](#_Toc14296)

[7. RecyclerView(常用) 60](#_Toc1038)

[8. RecyclerView(常用) 61](#_Toc27872)

[十、 系统服务 62](#_Toc16700)

[1.震动器 62](#_Toc29212)

[2.调节音量 62](#_Toc1616)

[3.定时服务 62](#_Toc813)

[3.位置服务 63](#_Toc5726)

[4.调用相机 65](#_Toc23777)

[十一、 开发小技巧 65](#_Toc28868)

[1. 解决启动白屏的问题 65](#_Toc1988)

[十二、 绘制与动画 66](#_Toc28669)

[1. 最基本的绘制示例 66](#_Toc20)

[2. 各种图形线条的绘制 67](#_Toc17460)

[3. 基础动画 70](#_Toc28313)

[4. 组合动画AnimationSet 71](#_Toc20304)

[5. 绘制圆形扩散特效果（实例） 72](#_Toc233)

[十三、 进阶功能实例 75](#_Toc22163)

[1.打包正式版本APK 75](#_Toc9506)

[2. 使用启动页 77](#_Toc24068)

[2.使用底部导航 78](#_Toc19488)

[5. 使用侧滑菜单DrawerLayout 80](#_Toc25651)

[6. 上拉刷新SwipeRefreshLayout 81](#_Toc11960)

[7. 高德地图 83](#_Toc10524)

[1）获取key 83](#_Toc13702)

[2） 文件的引入 84](#_Toc16696)

[3）引入后的对应配置 85](#_Toc24641)

[4）AndroidManifest.xml文件key及各权限的配置 86](#_Toc32251)

[5）初始化地图容器（layout） 87](#_Toc12478)

[6）初始化地图Activity 87](#_Toc16205)

[7）Amap地图控制器 87](#_Toc32524)

[7）比例尺（不生效） 88](#_Toc7615)

[6）绘点 88](#_Toc1857)

[7）绘线 88](#_Toc7061)

[8）沿着轨迹移动 88](#_Toc18172)

[9）在地图上设置功能按钮 89](#_Toc15057)

[10）离线地图 91](#_Toc12196)

[11）放大缩小的位置 92](#_Toc2276)

[12）去logo的方法 92](#_Toc7145)

[12）清除地图元素 92](#_Toc29882)

[13）定位到地图某个位置的两个方法 92](#_Toc10778)

[8. 高德地图的定位 93](#_Toc10800)

[1） AndroidManifest.xml的配置 93](#_Toc27203)

[2）activity\_main.xml的配置 94](#_Toc17977)

[3）MainActivity的配置 95](#_Toc1593)

[4）util/LocationUtil的配置 98](#_Toc4418)

[5）新版辅助H5页面定位 100](#_Toc31480)

[5）5.0版本以后定位写法 102](#_Toc6546)

[9. 高德地图的导航 103](#_Toc30209)

[1） 项目创建 103](#_Toc5553)

[2） 文件的引入AndroidManifest.xml文件key及各权限的配置 103](#_Toc16024)

[3） 文件的引入AndroidManifest.xml文件key及各权限的配置 104](#_Toc30504)

[10. 扫码（可做演示用） 105](#_Toc4634)

[11. 指纹识别 105](#_Toc6161)

# android sutdio的安装

## 1、下载安装（win64下安装3.1.2版本为例）

下载地址：<http://www.android-studio.org/>

或 [https://developer.android.google.cn/studio#downloads](https://developer.android.google.cn/studio" \l "downloads)

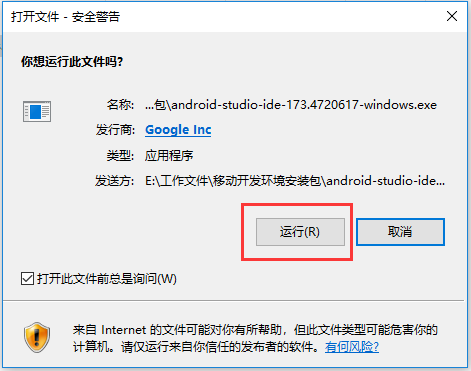
打开链接后有如下支持不同系统及系统位数的安装包：



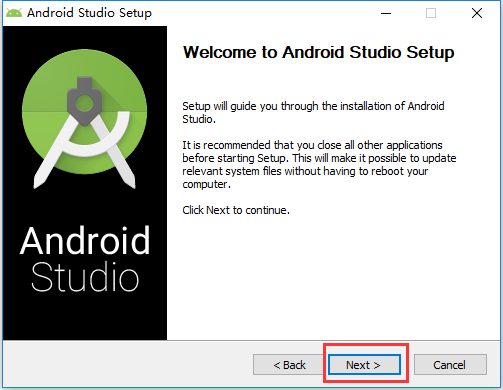
可以根据不同平台系统选择需要的android-studio安装包进行下载。

**安装步骤：**

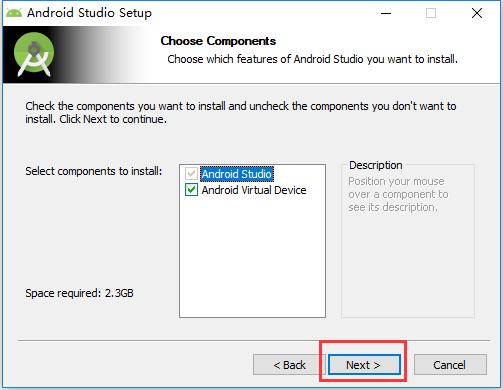
步骤1：双击打开



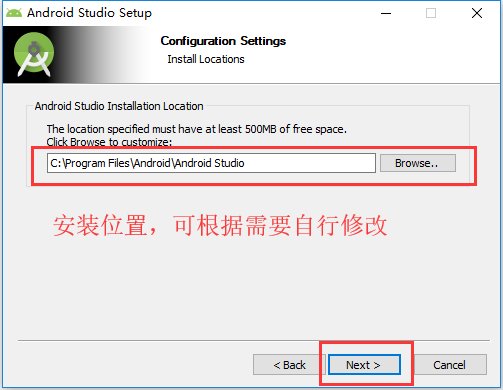
步骤2：点击运行（R）按钮



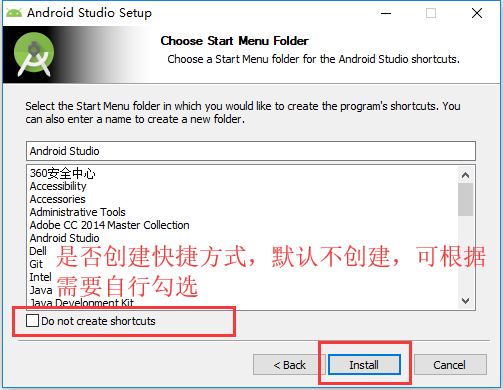
步骤3：点击next(下一步按钮)



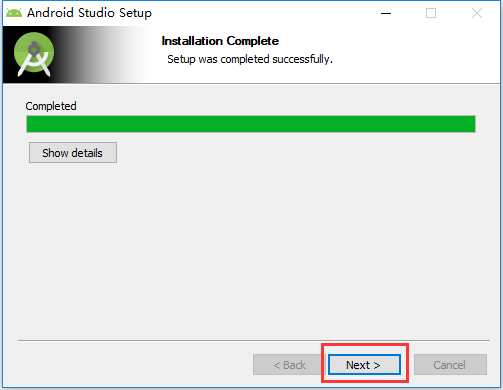
步骤4：设置安装位置，可根据需要自行修改，点击next(下一步) 按钮



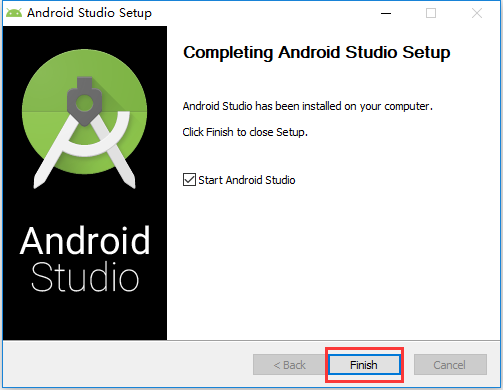
步骤5：选择是否需要创建快捷方式，默认不创建，可根据需要自行勾选，点击Install按钮进行安装。



步骤6：安装进度完成后点击Next(下一步)按钮



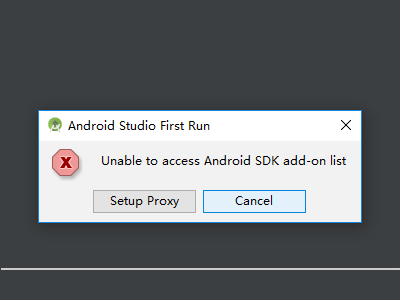
步骤7：点击finish按钮完成安装并关闭安装向导。



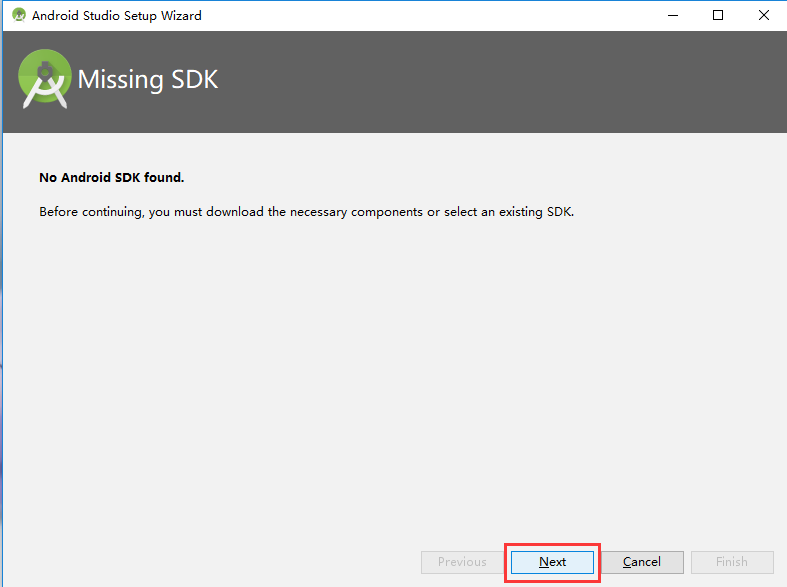
## 2、安装Android SDK(同时得先有JDK)

**安装步骤：**

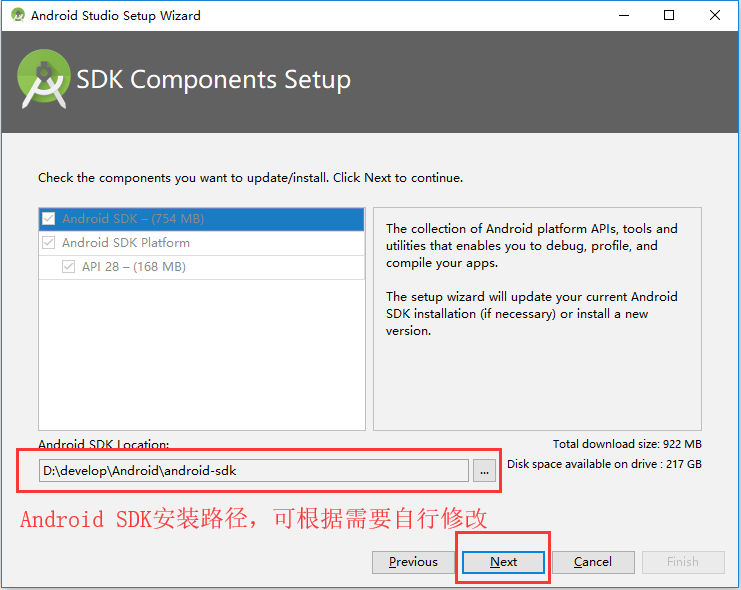
步骤1：打开4.1中安装的Android Studio,提示如下信息，点击cancel按钮



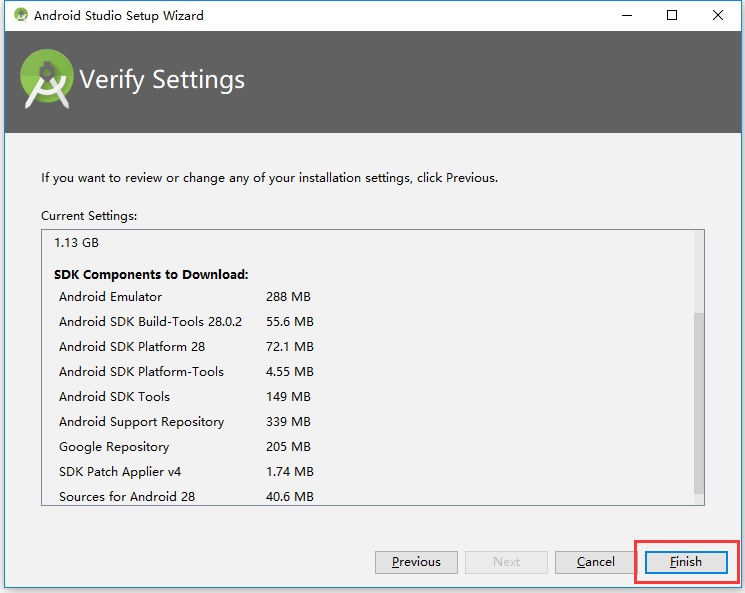
步骤2：点击Next（下一步）按钮



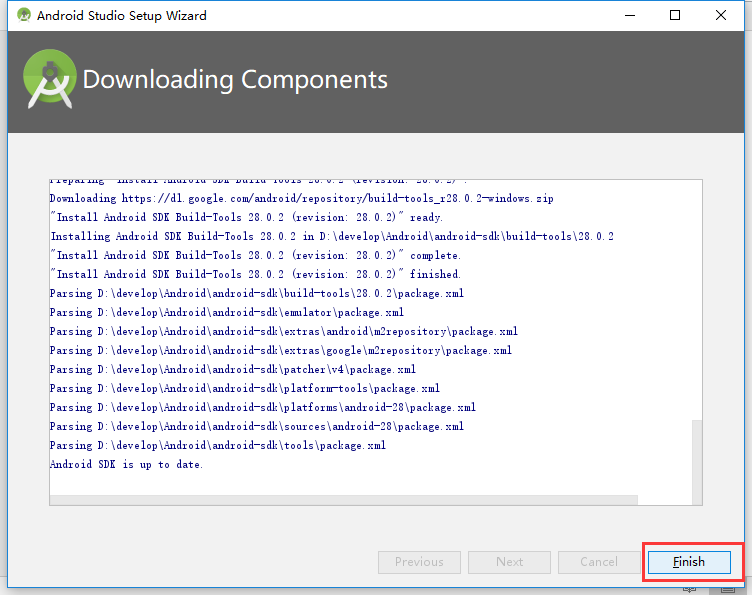
步骤3：选择Android SDK安装路径，点击Next（下一步）按钮

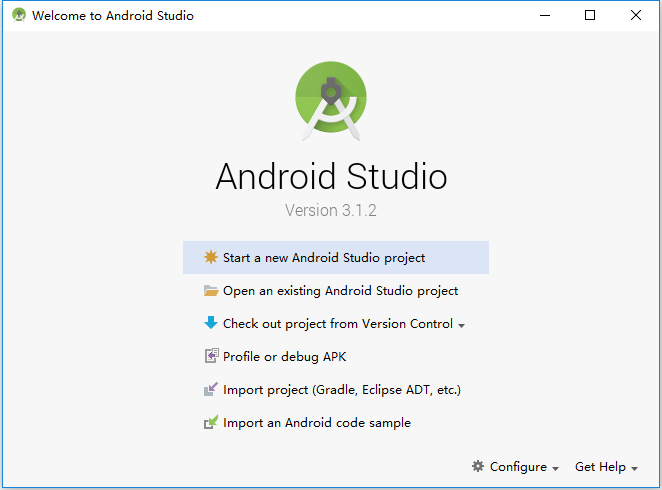


步骤4：点击Finish按钮开始安装Android SDK



步骤5：点击Finish按钮完成Android SDK安装

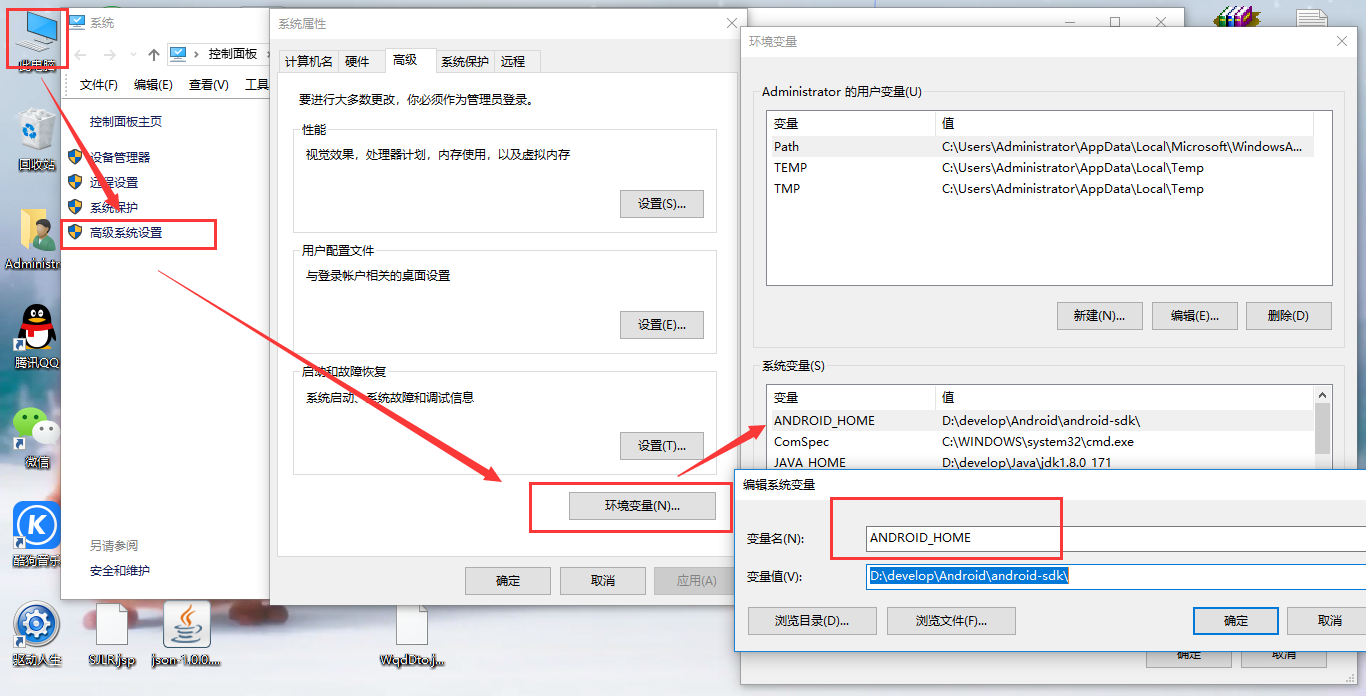




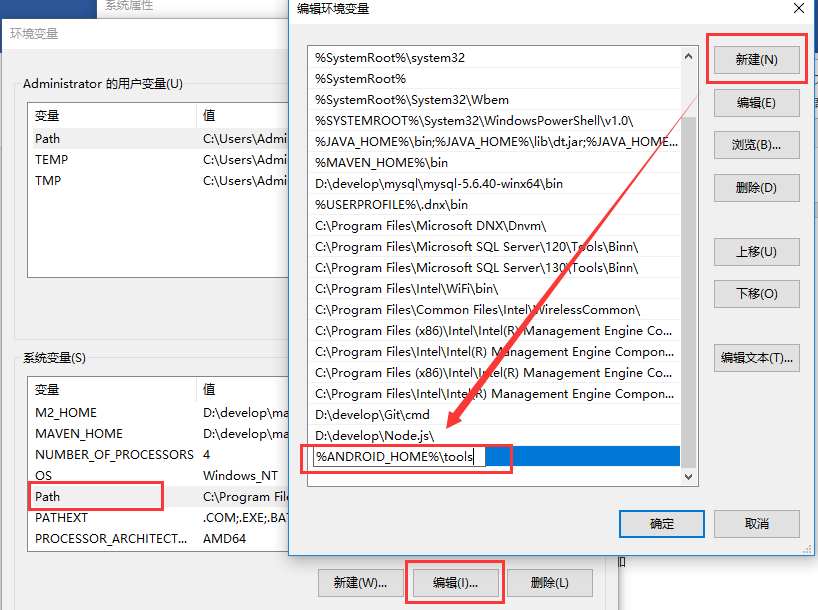
见如上界面则安装完成。

## 3、环境配置

点击计算机右键->属性->高级系统设置->环境变量->系统变量->新建进行配置，变量名：ANDROID\_HOME，变量值：android sdk安装路径，如：D:\develop\Android\android-sdk\



设置path，因为系统本来就有这个变量，点开这个变量，在变量值的最后加入;%ANDROID\_HOME%\tools 然后保存即可。



# 整体介绍

## 架构:基本附件组件

视图就是围绕着一下几个要素进行

|  |  |
| --- | --- |
| Fragments | 代表活动中的一个行为或者一部分用户界面。 |
| Views | 绘制在屏幕上的UI元素，包括按钮，列表等。 |
| Layouts | 控制屏幕格式，展示视图外观的View的继承。 |
| Intents | 组件间的消息连线。 |
| Resources | 外部元素，例如字符串资源、常量资源及图片资源等。 |
| Manifest | 应用程序的配置文件。 |

## 资源Resources

1. 资源机构

MyProject/

src/

MyActivity.java

res/

drawable/

icon.png

layout/

activity\_main.xml

info.xml

values/

strings.xml

1. 资源类型

| **目录** | **资源类型** |
| --- | --- |
| anim/ | 定义动画属性的XML文件。它们被保存在res/anim/文件夹下，通过R.anim类访问 |
| color/ | 定义颜色状态列表的XML文件。它们被保存在res/color/文件夹下，通过R.color类访问 |
| drawable/ | 图片文件，如.png,.jpg,.gif或者XML文件，被编译为位图、状态列表、形状、动画图片。它们被保存在res/drawable/文件夹下，通过R.drawable类访问 |
| layout/ | 定义用户界面布局的XML文件。它们被保存在res/layout/文件夹下，通过R.layout类访问 |
| menu/ | 定义应用程序菜单的XML文件，如选项菜单，上下文菜单，子菜单等。它们被保存在res/menu/文件夹下，通过R.menu类访问 |
| raw/ | 任意的文件以它们的原始形式保存。需要根据名为R.raw.filename的资源ID，通过调用Resource.openRawResource()来打开raw文件 |
| values/ | 包含简单值(如字符串，整数，颜色等)的XML文件。这里有一些文件夹下的资源命名规范。arrays.xml代表数组资源，通过R.array类访问；integers.xml代表整数资源，通过R.integer类访问；bools.xml代表布尔值资源，通过R.bool类访问；colors.xml代表颜色资源，通过R.color类访问；dimens.xml代表维度值，通过R.dimen类访问；strings.xml代表字符串资源，通过R.string类访问；styles.xml代表样式资源，通过R.style类访问 |
| xml/ | 可以通过调用Resources.getXML()来在运行时读取任意的XML文件。可以在这里保存运行时使用的各种配置文件 |

1. 资源使用

@color/opaque\_red

@string/hello

详细可参照开发-Res

## 活动Activity

1. 生命周期

| **回调** | **描述** |
| --- | --- |
| onCreate() | 这是第一个回调，在活动第一次创建时调用 |
| onStart() | 这个回调在活动为用户可见时被调用 |
| onResume() | 这个回调在应用程序与用户开始可交互的时候调用 |
| onPause() | 被暂停的活动无法接受用户输入，不能执行任何代码。当前活动将要被暂停，上一个活动将要被恢复时调用 |
| onStop() | 当活动不在可见时调用 |
| onDestroy() | 当活动被系统销毁之前调用 |
| onRestart() | 当活动被停止以后重新打开时调用 |

1. 生命周期实例

public class MainActivity extends Activity {

String msg = "Android : ";

/\*\* 当活动第一次被创建时调用 \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Log.d(msg, "The onCreate() event");

}

/\*\* 当活动即将可见时调用 \*/

@Override

protected void onStart() {

super.onStart();

Log.d(msg, "The onStart() event");

}

/\*\* 当活动可见时调用 \*/

@Override

protected void onResume() {

super.onResume();

Log.d(msg, "The onResume() event");

}

/\*\* 当其他活动获得焦点时调用 \*/

@Override

protected void onPause() {

super.onPause();

Log.d(msg, "The onPause() event");

}

/\*\* 当活动不再可见时调用 \*/

@Override

protected void onStop() {

super.onStop();

Log.d(msg, "The onStop() event");

}

/\*\* 当活动将被销毁时调用 \*/

@Override

public void onDestroy() {

super.onDestroy();

Log.d(msg, "The onDestroy() event");

}}

1. AndroidManifest.xml中声明

这个文件类似于route

<activity

android:name=".MainActivity"

android:label="@string/title\_activity\_main" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER"/>

</intent-filter>

</activity>

1. 开启单活动模式

这个文件类似于route

<activity android:name=".MainActivity"  
 android:launchMode="singleTask"  
 >  
 <intent-filter>

## 服务Service

建一个类继承service Myservers.java

package myser;  
import android.app.Service;  
import android.content.Intent;  
import android.os.Binder;  
import android.os.Handler;  
import android.os.IBinder;  
import android.os.Message;  
import android.util.Log;  
  
public class Myservers extends Service {  
 private static final String TAG="MyService";  
 //只有在生命周期第一次  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 Log.i(TAG,"onCreate");  
 }  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 Log.i(TAG,"onDestroy");  
 }  
 //每次startService都会被调用  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 Log.i(TAG,"onStartCommand ");  
 return super.onStartCommand(intent, flags, startId);  
 }  
 @Override  
 public boolean onUnbind(Intent intent) {  
 Log.i(TAG,"onUnbind ");  
 return super.onUnbind(intent);  
 }  
  
 //在整个生命周期中，只会被调用一次  
 @Override  
 public IBinder onBind(Intent intent) {  
 Log.i(TAG,"onBind");  
 return new MyBinder();  
 }  
 public class MyBinder extends Binder{  
 /\*  
 \* 计算平均值  
 \* @param scores 可变的数组，参数可以是0到N个  
 \* @return  
 \*/  
 public double calvAvg(double[] scores){  
 int count=scores.length;  
 if(count==0)  
 {  
 return 0;  
 }  
 double sum=0;  
 for(double s:scores){  
 sum+=s;  
 }  
 return sum/count;  
 }  
  
 private Thread thread=new mythreads();  
 public void startThread(){  
 thread.start();  
 }  
 }  
}  
  
class mythreads extends Thread{  
 @Override  
 public void run() {  
 super.run();  
  
 for (int i = 0; i <100 ; i++) {  
 //子线程不可更新UI线程中的控件的属性  
 Log.i("test",i+""+this.interrupted());  
 try {  
 this.sleep(1000); //过1秒执行一次  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

AndroidManifest.xml 中注册服务

<service android:name="myser.Myservers">  
 <intent-filter>  
 <action android:name="yargs.myservice"/>  
 </intent-filter>  
</service>

Xmld页面中定义几个按钮

<?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"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <Button  
 android:id="@+id/butstart"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="startService"/>  
  
 <Button  
 android:id="@+id/butstop"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="stopService"/>  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="50dp">  
 <Button  
 android:id="@+id/butbind"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="BindService"/>  
 <Button  
 android:id="@+id/butunbind"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="UnBindService"/>  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="50dp">  
 <Button  
 android:id="@+id/butsum"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="点这个按钮求和"  
 />  
 </LinearLayout>  
 </LinearLayout>  
</android.support.constraint.ConstraintLayout>

Activity中定义服务

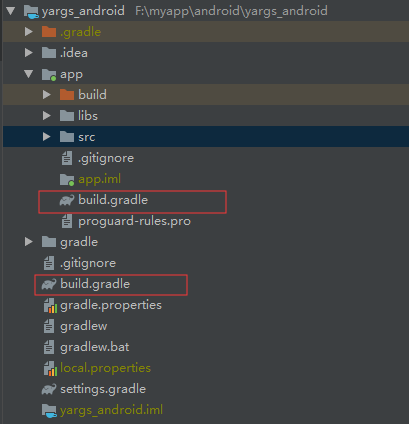
import android.content.ComponentName;  
import android.content.Intent;  
import android.content.ServiceConnection;  
import android.os.IBinder;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
  
import myser.Myservers;  
  
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
 private Button btnstart,btnstop;  
 private Button btnbind,btnunbind;  
 private Button btnSum;  
 private Intent intent;  
 private Myservers.MyBinder binder=null;  
 //匿名类  
 private ServiceConnection conn=new ServiceConnection() {  
 @Override  
 public void onServiceConnected(ComponentName name, IBinder service) {  
 //连接成功，自动调用  
 Log.i("Activity","onServiceConnected");  
 binder=(Myservers.MyBinder)service; //把binder绑定到server类里边  
 }  
  
 @Override  
 public void onServiceDisconnected(ComponentName name) {  
 //连接无效时调用  
 Log.i("Activity","onServiceDisconnected");  
 }  
 };  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 init();  
 intent = new Intent("yargs.myservice");  
 intent.setPackage(this.getPackageName()); //兼容Android 5.0  
 bindService(intent,conn,BIND\_AUTO\_CREATE); //bindService必须要有一个连接存在 BIND\_AUTO\_CREATE的意思是没有service自动创建一个  
 this.startService(intent);  
  
 }  
  
 //如果在Create中定义了bindservice，就在销毁方法中销毁bind  
 @Override  
 protected void onDestroy() {  
 unbindService(conn);  
 super.onDestroy();  
 }  
  
 private void init(){  
 btnstart=(Button)findViewById(R.id.butstart);  
 btnstop=(Button)findViewById(R.id.butstop);  
 btnstart.setOnClickListener(this);  
 btnstop.setOnClickListener(this);  
 btnbind=(Button)findViewById(R.id.butbind);  
 btnunbind=(Button)findViewById(R.id.butunbind);  
 btnbind.setOnClickListener(this);  
 btnunbind.setOnClickListener(this);  
 btnSum=(Button)findViewById(R.id.butsum);  
 btnSum.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 double [] sr={1.1,2.3,4.1,5.1};  
 double result=binder.calvAvg(sr);  
 binder.startThread();  
 //连接无效时调用  
 Log.i("result",result+"");  
 }  
 });  
 }  
  
 @Override  
 public void onClick(View v) {  
  
 switch (v.getId()){  
 case R.id.butstart:  
 intent = new Intent("yargs.myservice");  
 intent.setPackage(this.getPackageName()); //兼容Android 5.0  
 this.startService(intent);  
 break;  
 case R.id.butstop:  
 intent=new Intent("yargs.myservice");  
 intent.setPackage(this.getPackageName()); //兼容Android 5.0  
 this.stopService(intent);  
 break;  
 case R.id.butbind:  
 intent = new Intent("yargs.myservice");  
 intent.setPackage(this.getPackageName()); //兼容Android 5.0  
 bindService(intent,conn,BIND\_AUTO\_CREATE); //bindService必须要有一个连接存在 BIND\_AUTO\_CREATE的意思是没有service自动创建一个  
 this.startService(intent);  
 break;  
 case R.id.butunbind:  
// intent = new Intent("yargs.myservice");  
// intent.setPackage(this.getPackageName()); //兼容Android 5.0  
//  
// this.startService(intent);  
 this.unbindService(conn);  
 break;  
  
  
 }  
 }  
}

## AndroidManifest.xml详解

<?xml version="1.0" encoding="utf-8"?>  
<!--根元素 manifest  
package 包名 应用程序的唯一标识  
android:versionCode 版本号  
android:versionName 版本名  
-->  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.demo02.myapplication"  
 android:versionCode="1"  
 android:versionName="1.0">  
 <!--sdk最小版本 当前版本-->  
 <uses-sdk  
 android:minSdkVersion="8"  
 android:targetSdkVersion="15"/>  
 <!--application中是对当前应用程序组件进行说明的地方  
 label标题栏显示的文字  
 -->  
 <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">  
 <!--用activity来说明组件  
 name 是类名 可以写全名 com.example.demo02.myapplication.MainActivity 也可以只写.MainActivity  
 label是标签 是标题栏的名称 有就显示，没有就显示外边公共公共定义的label  
 -->  
 <activity android:name=".MainActivity">  
 </activity>  
 <activity android:name=".Main2Activity"  
 android:label="我是这个页面的名称">  
  
 <!--用下边来定义启动项-->  
 <intent-filter>  
 <!--逻辑动作名-->  
 <action android:name="android.intent.action.MAIN" />  
 <!--类别，是一个启动项-->  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 </activity>  
 </application>  
</manifest>

## build.gradle详解

项目中一般会出现2个或者多个 build.gradle 文件，一个在根目录下，一个在 app 目录下，如果有模块，模块中又分别有一个



主项目里边的一般只配置gradle

dependencies {  
 classpath 'com.android.tools.build:gradle:3.5.2'  
}

compileSDKVersion 用于指定项目的变异 SDK  版本

android 闭包，配置项目构建的各种属性，

◆compileSDKVersion 用于指定项目的变异 SDK  版本，

◆buildToolsVersion 用户指定项目构建工具的版本。

◆ defaultConfig 闭包：默认配置、应用程序包名、最小sdk版本、目标sdk版本、版本号、版本名称；

◆ buildTypes 闭包：指定生成安装文件的配置，是否对代码进行混淆；

◆ signingConfigs 闭包：签名信息配置；

◆ sourceSets  闭包：源文件路径配置；

◆ lintOptions 闭包：lint 配置；

dependencies 闭包，指定当前项目的所有以来关系，本地以来，库依赖以及远程依赖；

repositories 闭包，仓库配置。

实际使用中，最多的就是引用一下包

dependencies {  
 implementation fileTree(dir: 'libs', include: ['\*.jar'])  
 androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
}

build.gradle全解

// 声明是Android程序，

// com.android.application 表示这是一个应用程序模块,可直接运行

// com.android.library 标识这是一个库模块，是依附别的应用程序运行

apply plugin: 'com.android.application'

android {

// 程序在编译的时候会检查lint，有任何错误提示会停止build，我们可以关闭这个开关

lintOptions {

// 即使报错也不会停止打包

abortOnError false

// 打包release版本的时候是否进行检测

checkReleaseBuilds false

}

// 编译sdk的版本，也就是API Level，例如API-19、API-20、API-21等等。

compileSdkVersion 27

// build tools的版本，其中包括了打包工具aapt、dx等等。

// 这个工具的目录位于你的sdk目录/build-tools/下

buildToolsVersion '27.0.3'

//关闭Android Studio的PNG合法性检查

aaptOptions.cruncherEnabled = false

aaptOptions.useNewCruncher = false

defaultConfig { // 默认配置

applicationId "com.demo.test" // 应用程序的包名

minSdkVersion 22 / 最小sdk版本，如果设备小于这个版本或者大于maxSdkVersion将无法安装这个应用

targetSdkVersion 27 // 目标sdk版本，充分测试过的版本（建议版本）

versionCode 1 // 版本号，第一版是1，之后每更新一次加1

versionName "1.0" // 版本名，显示给用户看到的版本号

archivesBaseName = "demo-$versionName" // 指定打包成Jar文件时候的文件名称

ndk {

moduleName "testwifisafe" // 设置库(so)文件名称

ldLibs "log", "z", "m", "jnigraphics", "android"

// 引入库，比如要用到的\_\_android\_log\_print

abiFilters "armeabi", "x86", "armeabi-v7a" // "x86" 显示指定支持的ABIs

cFlags "-std=c++11 -fexceptions" // C++11

stl "gnustl\_static"

}

// 当方法数超过65535(方法的索引使用的是一个short值，

// 而short最大值是65535)的时候允许打包成多个dex文件，动态加载dex。这里面坑很深啊

multiDexEnabled true

// Instrumentation单元测试

testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

}

// 默认的一些文件路径的配置

sourceSets {

main {

manifest.srcFile 'AndroidManifest.xml' // 指定清单文件

res.srcDirs = ['res'] // 指定res资源目录

assets.srcDirs = ['assets'] // asset资源目录

jni.srcDirs 'src/main/jni' // jni代码目录

jniLibs.srcDir 'src/main/jniLibs' // jni库目录

java.srcDirs = ['src'] // 指定java源代码目录

resources.srcDirs = ['src'] // 指定resource目录

aidl.srcDirs = ['src'] // 指定aidl目录

renderscript.srcDirs = ['src'] // 指定source目录

}

debug.setRoot('build-types/debug') // 指定debug模式的路径

release.setRoot('build-types/release') // 指定release模式的路径

}

// multiDex的一些相关配置，这样配置可以让你的编译速度更快

dexOptions {

// 让它不要对Lib做preDexing

preDexLibraries = false

// 开启incremental dexing,优化编译效率，这个功能android studio默认是关闭的。

incremental true

javaMaxHeapSize "4g" // 增加java堆内存大小

}

signingConfigs {// 签名配置

release {// 发布版签名配置

storeFile file("fk.keystore") // 密钥文件路径

storePassword "123456" // 密钥文件密码

keyAlias "fk" // key别名

keyPassword "123456" // key密码

}

debug {// debug版签名配置

storeFile file("fk.keystore")

storePassword "123456"

keyAlias "fk"

keyPassword "123456"

}

}

// 指定生成安装文件的配置，常有两个子包:release,debug，注：直接运行的都是debug安装文件

buildTypes {

// release版本的配置，即生成正式版安装文件的配置

release {

zipAlignEnabled true // 是否支持zip

shrinkResources true // 移除无用的resource文件

minifyEnabled false // 是否对代码进行混淆，true表示混淆

// 指定混淆时使用的规则文件；

// proguard-android.txt指所有项目通用的混淆规则，proguard-rules.pro当前项目特有的混淆规则

// release的Proguard默认为Module下的proguard-rules.pro文件

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

debuggable false //是否支持调试

//ndk的一些配置

ndk {

// cFlags "-std=c++11 -fexceptions -O3 -D\_\_RELEASE\_\_" // C++11

// platformVersion = "19"

moduleName "hebbewifisafe" // 设置库(so)文件名称

ldLibs "log", "z", "m", "jnigraphics", "android"

// 引入库，比如要用到的\_\_android\_log\_print

abiFilters "armeabi", "x86", "armeabi-v7a"// "x86"

cFlags "-std=c++11 -fexceptions" // C++11

stl "gnustl\_static"

}

// 采用动态替换字符串的方式生成不同的 release.apk （3.0之后版本的修改方式）

applicationVariants.all { variant ->

variant.outputs.all { output ->

if (!variant.buildType.isDebuggable()) {

// 获取签名的名字 variant.signingConfig.name

// 要被替换的源字符串

def sourceFile = "app-release";

// 替换的字符串

　　　　　　　　　　　　　　def replaceFile = "Demo-V${variant.versionName}.${releaseTime()}"

　　　　　　　　　　　　　　outputFileName = output.outputFile.name.replace(sourceFile, replaceFile)

}

}

}

jniDebuggable false // 关闭jni调试

}

debug { // debug版本的配置

minifyEnabled false

zipAlignEnabled true

shrinkResources true // 移除无用的resource文件

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

debuggable true

// jniDebuggable true

ndk {

cFlags "-std=c++11 -fexceptions -g -D \_\_DEBUG\_\_" // C++11

}

jniDebuggable true

}

}

packagingOptions {

exclude 'META-INF/ASL2.0'

exclude 'META-INF/LICENSE'

exclude 'META-INF/NOTICE'

exclude 'META-INF/MANIFEST.MF'

}

compileOptions {

// 在这里你可以进行 Java 的版本配置，

// 以便使用对应版本的一些新特性

}

productFlavors {

// 在这里你可以设置你的产品发布的一些东西，

// 比如你现在一共软件需要发布到不同渠道，

// 且不同渠道中的包名不同，那么可以在此进行配置；

// 甚至可以设置不同的 AndroidManifest.xml 文件。

hebbe {

}

googlePlay {

}

solo {

}

}

productFlavors.all {

flavor -> flavor.manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: name]

}

// 所谓ProductFlavors其实就是可定义的产品特性，

// 配合 manifest merger 使用的时候就可以达成在一次编译

// 过程中产生多个具有自己特性配置的版本。

// 上面这个配置的作用就是，为每个渠道包产生不同的 UMENG\_CHANNEL\_VALUE 的值。

}

// 指定当前项目的所有依赖关系：本地依赖、库依赖、远程依赖

// 本地依赖：可以对本地 Jar 包或目录添加依赖关系

// 库依赖：可以对项目中的库模块添加依赖关系

// 远程依赖：可以对 jcenter 库上的开源项目添加依赖

// 标准的远程依赖格式是 域名:组织名:版本号

dependencies {

implementation fileTree(dir: 'libs', include: ['\*.jar']) // 本地依赖

// 远程依赖，com.android.support是域名部分，appcompat-v7是组名称，26.1.0是版本号

implementation 'com.android.support:appcompat-v7:26.1.0'

implementation 'com.android.support.constraint:constraint-layout:1.0.2'

implementation project(':hello') // 库依赖

testImplementation 'junit:junit:4.12' // 声明测试用列库

androidTestImplementation 'com.android.support.test:runner:1.0.1'

androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'

}

// 声明是要使用谷歌服务框架

apply plugin: 'com.google.gms.google-services'

// 第三方依赖库的本地缓存路径

task showMeCache << {

configurations.compile.each { println it }

}

// 使用maven仓库。android有两个标准的library文件服务器，一个jcenter一个maven。两者毫无关系。

// jcenter有的maven可能没有，反之亦然。

// 如果要使用jcenter的话就把mavenCentral()替换成jcenter()

repositories {

mavenCentral()

}

205 // 获取日期方法

def releaseTime() {

return new Date().format("MMdd.HHmm")

}

# 开发

## 赋值取值

赋值：setText

EditText edittext1= (EditText) findViewById(R.id.edit1);  
edittext1.setText("赋予的值");

取值：getText

EditText edittext1= (EditText) findViewById(R.id.edit1);  
String aa=edittext1.getText().toString();

## 点击事件

【直接绑定法】

Xml:android:onClick="方法名"

Activity

import android.view.View;

public void 方法(View view) {

}

视图插件必须引入，不然这个事件会退出系统

【通过ID绑定法】

在onCreate方法里

Button button1=(Button) findViewById(R.id.button1);

button1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(MainActivity.this,"进了myClick方法",Toast.LENGTH\_LONG).show();

}

});

备注：方法中用到的this,如过是通过监听执行的方法，这里需要activity名+this

## 主题及颜色

配置主题：AndroidManifest.xml

android:theme="@style/AppTheme"

定义的这个主题来自于res/values/styles.xml

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<!--Appbar背景色-->

<item name="colorPrimary">@color/colorPrimary</item>

<!--状态栏颜色-->

<item name="colorPrimaryDark">@color/colorPrimaryDark</item>

<!--是否显示title，如果设置了windowActionBar = false则必须设置windowNoTitle = true，否则会报错-->

<item name="windowNoTitle">true</item>

<item name="windowActionBar">false</item>

<!--设置遮盖（activity启动时短暂展示的画面）-->

<item name="android:windowContentOverlay">@null</item>

<!--覆盖模式（需要手动设置布局的paddingTop或者marginTop，确保布局在actionbar的下方）-->

<item name="android:windowActionBarOverlay">true</item>

<!--Button正常状态下的颜色-->

<item name="colorButtonNormal">@color/colorAccent</item>

<!--控件文字的颜色（例如button）-->

<item name="android:textColorPrimary">@color/holo\_orange\_light</item>

<!--各个控制控件的默认颜色-->

<item name="colorControlNormal">@color/colorPrimary</item>

<!--控制各个控件被选中时的颜色-->

<item name="colorAccent">@color/colorAccent</item>

<!--控件点击波纹效果的颜色（适用于5.0以上）-->

<item name="colorControlHighlight">@color/btn\_blue\_normal</item>

<!--页面背景色-->

<item name="android:windowBackground">@color/holo\_red\_light</item>

</style>

如果想定义其它蓝色，paernt=”Theme.AppCompat.Light.DarkActionBar”

设置主题是在 AndroidManifest.xml中

更详细的切换主题功能查看后文 主题切换

主题中定义的颜色来自于res/values/colors.xml

<resources>  
 <color name="colorTitle">#ff4f02</color>  
 <color name="colorNav">#ccc</color>  
</resources>

## Res、图片资源、menu

对着res点右键 new-Android Resource File 可以新建各种类型的资源并创建文件夹

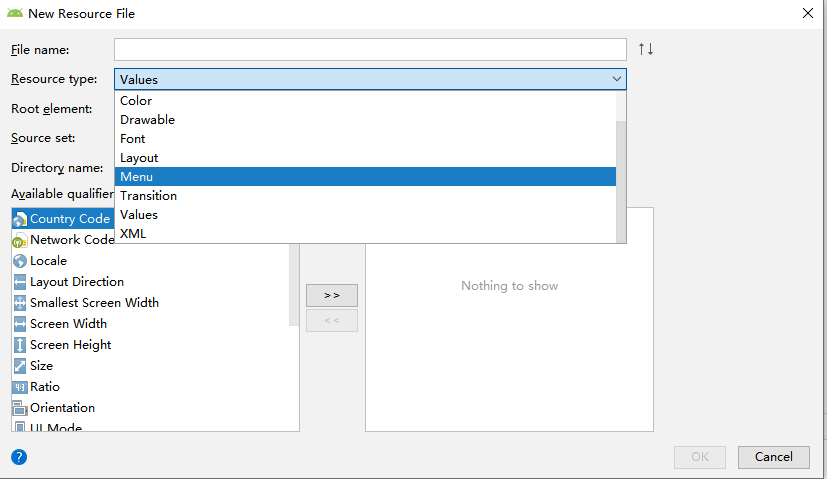


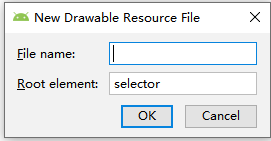
Image Asset和 Vector Asset对着res点右键和对着drawable点右键效果一样该全局生成全局生成，该生成单个文件就生成在drawable文件夹下的单个文件

new - image Asset 这个创建的是兼容各种大小的图片，建一次生成多个

new - Vector Asset 这个创建的是单个svg图片，这个较为常用

## 选中和未选中表现为不同图片不同颜色

drawable点右键 new - Drawable resource file

输入名称 不需要带.xml，后边会自动生成

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_checked="true" android:drawable="@drawable/nav1"/>

<item android:state\_checked="false" android:drawable="@drawable/nav2"/>

</selector>

上边是定义显示不同的两个图片

还可以定义不同的颜色 (注：这个是在drawable里边注册的颜色)

<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
 <item android:color="#3ab921" android:state\_checked="true" /><!--点击显示-->  
 <item android:color="#e758ca" android:state\_checked="false" /><!--默认显示-->  
</selector>

如果使用 BottomNavigationView 等这些控件，加以下两个参数

app:itemIconTint="@drawable/nav\_choose\_color" // 选中图标的颜色  
app:itemTextColor="@drawable/nav\_choose\_color" // 选中文字的颜色

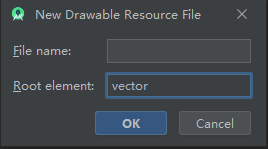
可配合下边使用svg图片

## 如何使用svg图片

Iconfont 或者自己用ai做一个svg 图片，打开下边网址，并拖入其中

<http://inloop.github.io/svg2android/>

drawable点右键 new - Drawable resource file

输入名称 不需要带.xml，后边会自动生成

直接把网址中生成的代码全部复制到xml中，调用即可

## menu的结构内容

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

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/navigation\_home"

android:icon="@drawable/ic\_home\_black\_24dp"

android:title="首页" />

<item

android:id="@+id/navigation\_dashboard"

android:icon="@drawable/ic\_car"

android:title="出行" />

</menu>

## 设置图片路径 及 ScaleType属性

ImageView a=(ImageView) findViewById(R.id.ss);

a.setImageResource(R.drawable.s);

ScaleType属性

| 常量 | 含义 |
| --- | --- |
| fitXY | 横向、纵向独立缩放，以适应该ImageView |
| fitCenter | 保持纵横比缩放图片，缩放完成后将图片放在ImageView的中央 |
| fitStart | 保持纵横比缩放图片，并且将图片放在ImageView的左上角 |
| fitEnd | 保持纵横比缩放图片，缩放完成后将图片放在ImageView的右下角 |
| center | 把图片放在ImageView的中央，但是不进行任何缩放 |
| centerCrop | 保持纵横比缩放图片，以使图片能完全覆盖ImageView |
| centerInside | 保持纵横比缩放图片，以使得ImageView能完全显示该图片 |
| matrix | 使用matrix方式进行缩放 |

## 多语言版本

把strings.xml 复制到另一个文件夹中，改显示的值

values-zh 设置中文

values 这个是默认英文

## json的处理

定义json

方法一

private String json = "{\"name\",\"张三\"}";

方法二 res-raw下建.json文件

{  
 **"name"**:**"张三"**,  
 **"age"**:17,  
 **"address"**:{  
 **"city"**:**"中国"**,  
 **"state"**:**"CN"** },  
 **"dataList"**:  
 [  
 {**"company"**:**"taipingyang"**},  
 {**"company"**:**"taipingyang1"**},  
 {**"company"**:**"taipingyang2"**},  
 {**"company"**:**"taipingyang3"**},  
 {**"company"**:**"taipingyang4"**},  
 {**"company"**:**"taipingyang5"**}  
 ]  
}

如果是定义的变量json,可以直接获取

**try** {  
 JSONObject jsonObject = **new** JSONObject(json);  
 *//通过key得到value* String value = jsonObject.getString(**"name"**);  
 Log.*i*(**"text"**, value);  
} **catch** (JSONException e) {  
 e.printStackTrace();  
}

如果是.json文件，固定写法通过流转1024字节得到json

**try** {  
 String data =getConent(getResources(),R.raw.***jsonp***);  
 JSONObject jsonObject = **new** JSONObject(data);  
 *//通过key得到value* String name = jsonObject.getString(**"name"**);  
 Log.*i*(**"text"**, name);  
  
 *//通过key获取对象，再获取值* JSONObject address=jsonObject.getJSONObject(**"address"**);  
 String city = address.getString(**"city"**);  
 Log.*i*(**"text"**, city);  
  
 JSONArray dataList=jsonObject.getJSONArray(**"dataList"**);  
 **for**(**int** index=0;index<dataList.length();index++)  
 {  
 JSONObject obj=dataList.getJSONObject(index);  
 String company=obj.getString(**"company"**);  
 }  
 JSONObject obj2=dataList.getJSONObject(2);  
 String company2=obj2.getString(**"company"**);  
 Log.*i*(**"text"**, company2);  
  
} **catch** (Exception e) {  
 e.printStackTrace();  
}

固定写法通过流转1024字节，后边可以写成公共方法

*/\*  
 \* 得到json文件中的内容  
 \* @parm res  
 \* @id  
 \* @return  
 \* @throws  
 \*/***private** String getConent(Resources res, **int** id) **throws** IOException {  
 InputStream is = **null**;  
 StringBuffer sb = **new** StringBuffer();  
 **try** {  
 is = res.openRawResource(id);  
 **byte**[] buffer = **new byte**[1024];  
 **int** len = is.read(buffer, 0, 1024);  
 **while** (len != -1) {  
 String s = **new** String(buffer, 0, len);  
 sb.append(s);  
 len = is.read(buffer, 0, 1024);  
 }  
 } **finally** {  
 **if** (is != **null**) {  
 is.close();  
 }  
 }  
 **return** sb.toString();  
}

## 渐变遮罩

Drawable/建一个xml文件，调用以调图片的形式

这个的原理是做一张有不透明度的图片，放在另一个图片之上

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

<shape xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_height="wrap\_content" android:layout\_width="wrap\_content">

<gradient

android:angle="270"

android:endColor="@android:color/transparent"

android:startColor="#CC000000" />

<!-- shape节点中, 可以通过android:shape来设置形状, 默认是矩形.

gradient节点中angle的值270是从上到下，0是从左到右，90是从下到上。

此处的效果就是从下向上, 颜色逐渐由纯透明慢慢变成黑透色-->

</shape>

这个的原理是做一张有不透明度的图片，放在另一个图片之上

<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <gradient  
 android:angle="135"  
 android:centerColor="#009688"  
 android:endColor="#00695C"  
 android:startColor="#4DB6AC"  
 android:type="linear" />  
</shape>

**android:angle**

Integer，代表渐变颜色的角度， 0 is left to right, 90 is bottom to top. 必须是45的整数倍.

默认是 0.该属性只有在type=linear情况下起作用，默认的type为linear。

**android:startColor**

Color. 颜色渐变的开始颜色，如angle=270中的 android:startColor="#000000"

**android:endColor**

Color. 颜色渐变的结束颜色，如angle=270中的  android:endColor="#ffffff"

**android:centerX**

Float.(0 - 1.0) 相对X的渐变位置。

**android:centerY**

Float.(0 - 1.0) 相对Y的渐变位置。

这两个属性只有在type不为linear情况下起作用。

android:gradientRadius

Float. 渐变颜色的半径，单位应该是像素点. 需要 android:type="radial".

    如果android:type="radial"，没有设置android:gradientRadius，将会报错，error inflating class.

android:type

|  |  |
| --- | --- |
| Value | Description |
| "linear" | 线性渐变.可以理解为 y=kx+b. |
| "radial" | A radial gradient.圆形渐变，起始颜色从cenralX,centralY点开始。 |
| "sweep" | A sweeping line gradient. |

# 基础组件及布局

## 重绘标题栏ToolBar

android虽然系统自带标题栏，可很多时候标题栏需要自定义，很多时候标题栏又遮住了其它控件，大部分时候，都需要重绘标题栏

### 样式隐藏，代码绑定

首先设置样式

Res/values/style.xml中添加一类样式

<style name="AppTheme.NoActionBar">  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
</style>

在AndroidManifest.xml中配置的activity设置标题不显示的样式

<activity  
 android:name=".ScrollingActivity"  
 android:label="@string/app\_name"  
 android:theme="@style/AppTheme.NoActionBar">

注意：theme主题只能用android:theme=”@style/AppTheme （appTheme主题或者其子类），而不能用android:style。 否则会提示错误： Caused by: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.

通常情况下，隐藏标题栏不是不要标题栏，而是根据自己的需要重绘标题栏

androidx的写法

<androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>

android的写法

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="@color/colorPrimary"

app:logo="@mipmap/ic\_launcher"

app:title="标题"

app:titleTextColor="#fff"

app:subtitle="副标题"

app:subtitleTextColor="#fff"

app:navigationIcon="@drawable/ic\_menu"

android:theme="@style/Base.Theme.AppCompat.Light"

app:popupTheme="@style/toolBar\_pop\_item"

>

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

((AppCompatActivity) getActivity()).setSupportActionBar(toolbar);

### 代码隐藏，前端重绘

代码隐藏

ActionBar actionbar=getSupportActionBar();  
if(actionbar!=null) actionbar.hide();

前端重绘

>LinearLayout

>Toolbar

>ImageView+TextView

>其它控件

## 状态栏

需要activity在onCreate的时候才可以设置

getWindow().addFlags(WindowManager.LayoutParams.ALPHA\_CHANGED);//隐藏状态栏

半透明

if(Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.KITKAT) {  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);  
}

如果冲毁标题栏，且设置完半透明后

<androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="92dp"  
 android:background="@drawable/titlebg"  
 tools:ignore="MissingConstraints">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:paddingTop="20dp"  
 android:text="yargs"  
 android:textSize="28sp"  
 android:textColor="@color/ColorWrite" />  
</androidx.appcompat.widget.Toolbar>

## 组件库AndroidX

Google 2018 IO 大会推出了 Android新的扩展库 AndroidX，用于替换原来的 Android扩展库，将原来的android.\*替换成androidx.\*；只有包名和Maven工件名受到影响，原来的类名，方法名和字段名不会更改。接下来我们来看看使用 AndroidX的扩展库需要哪些配置。

全引用

implementation 'androidx.appcompat:appcompat:1.1.0'

implementation 'com.google.android.material:material:1.0.0'

implementation 'androidx.legacy:legacy-support-v4:1.0.0'

implementation 'androidx.recyclerview:recyclerview:1.0.0'

implementation 'androidx.constraintlayout:constraintlayout:1.1.3'

Fragment 如果需要单独引用的话，一般已经有了，不需要再引用

implementation 'androidx.fragment:fragment:1.1.0'

**ConstraintLayout** 注意：这个是包名都改了

androidx.coordinatorlayout.widget.CoordinatorLayout

android.support.constraint.ConstraintLayout

这个很重要，很多插件都必须放里边

**Toolbar**

androidx.appcompat.widget.Toolbar

android.support.v7.widget.Toolbar

**DrawerLayout**

androidx.drawerlayout.widget.DrawerLayout

android.support.v4.widget.DrawerLayout

**BottomNavigationView**

com.google.android.material.bottomnavigation.BottomNavigationView

android.support.design.widget.BottomNavigationView

**ViewPager**

androidx.viewpager.widget.ViewPager

android.support.v4.view.ViewPager

**AppBarLayout**

com.google.android.material.appbar.AppBarLayout

android.support.design.widget.AppBarLayout

**CollapsingToolbarLayout**

com.google.android.material.appbar.CollapsingToolbarLayout

android.support.design.widget.CollapsingToolbarLayout

**NestedScrollView**

androidx.core.widget.NestedScrollView

android.support.v4.widget.NestedScrollView

## 弹框Toast/Dialog/Snackbar

1. 三者的异同

Snackbar和Dialog、Toast一样都是用来作为android内提示信息的，三者之间的应用场景也有所不同。

【**Dialog**】

模态对话框。也就说，此刻该对话框中的内容获取了焦点，想要操作对话框以外的功能，必须先对该对话框进行响应。

应用场景：对于删除确认、版本更新等重要性提示信息，需要用户做出选择的情况下，使用Dialog。

【**Toast**】

非模态提示框。也就说提示框的显示并不影响我们对其他地方的操作，Toast无法手动控制隐藏，需要设置Toast的显示时长，一旦显示时间结束，Toast会自动消失。如果多次点击并显示Toast，就会出现Toast重复创建并显示，给用户造成一种Toast长时间不隐藏的幻觉。

应用场景：对于无网络提示、删除成功、发布操作完成等这类不重要的提示性信息，使用Toast；

【**Snackbar**】

Snackbar和Toast比较相似，但是用途更加广泛，并且它是可以和用户进行交互的。Snackbar使用一个动画效果从屏幕的底部弹出来，过一段时间后也会自动消失。

应用场景：删除操作时，弹出Snackbar用于确认删除操作；消息发送失败时，弹出Snackbar，用于重新发送操作；当然重要的是与MD组件相结合，用户体验效果更佳。

1. Toast

第一个参数是本activity，第二个是要传的中文，第二个是时长，也可用Toast.LENGTH\_LONG等

Toast.makeText(this,"这是xml方式指定的Button响应",Toast.LENGTH\_LONG).show();

//普通的显示方法  
Toast.makeText(this,"我是一个toast",Toast.LENGTH\_LONG).show();  
  
//通过构造函数来显示，setView只能传一次，要么是图片，要么是文字  
Toast toast=new Toast(this);  
ImageView img=new ImageView(this);  
img.setBackgroundResource(R.drawable.ic\_android\_black\_24dp);  
TextView textView=new TextView(this);  
textView.setText("sssssssss");  
toast.setView(textView);  
toast.setView(img);  
toast.setDuration(Toast.LENGTH\_LONG);  
toast.show();  
  
// 这种方法是给toast赋予一个页面，这个页面就可以很复杂

//建一个xml文件 toastview，里边有简单的内容

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content">  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 app:srcCompat="@mipmap/ic\_launcher\_round" />  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="我是一个复杂的Toast" />  
</LinearLayout>

View totatView=LayoutInflater.from(this).inflate(R.layout.toastview,null);  
Toast toast2=new Toast(this);  
toast2.setView(totatView);  
toast2.setDuration(Toast.LENGTH\_LONG);  
toast2.setGravity(Gravity.CENTER,0,0);  
toast2.show();

1. AlertDialog

//显示经典对话框  
AlertDialog dlg = new AlertDialog.Builder(this)  
 .setIcon(R.drawable.ic\_android\_black\_24dp)  
 .setTitle("提示信息") //标题和图片都不设，会不显示对话框的标题栏  
 .setMessage("这是一个测试对话框")  
 .setNeutralButton("写什么", null) //最左边的按钮  
 .setPositiveButton("确定", null) //最右边靠左边的按钮  
 .setNegativeButton("取消", null) //最右边的按钮  
 .create();  
dlg.show();

//显示带方法  
new AlertDialog.Builder(this)  
 .setMessage("这是一个测试对话框")  
 .setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.makeText(MainActivity.this,"点了确定",Toast.LENGTH\_LONG).show();  
 }  
 })  
 .setNegativeButton("取消", null) //最右边的按钮  
 .show();

1. Snackbar

Snackbar.make(view, message, duration)

.setAction(action message, click listener)

.show();

message显示的内容，一般只能显示2行；

duration是Snackbar想要显示的时间长短，有三个值：LENGTH\_INDEFINITE 永远显示、LENGTH\_LONG显示较长时间、LENGTH\_SHORT 显示较短时间；

示例

Snackbar.make(view, "要显示的文字", Snackbar.LENGTH\_INDEFINITE)

.setAction("Action", null).show();

## 滚动插件ScrollView/HorizontalScrollV

<ScrollView>

<HorizontalScrollView>

横向与纵向滚动条只需要注意里边只能有一个容器就可以了

## 浮动按钮FloatingActionButton

android:src：FAB中显示的图标.

app:backgroundTint：正常的背景颜色 ，这里是ColorStateList类型

app:rippleColor：按下时的背景颜色

app:elevation：正常的阴影大小

app:pressedTranslationZ：按下时的阴影大小

app:layout\_anchor：设置FAB的锚点，即以哪个控件为参照设置位置

app:layout\_anchorGravity：FAB相对于锚点的位置

app:fabSize：FAB的大小，normal或mini（分别对应56dp和40dp）

app:borderWidth：边框大小，最好设置成0dp否则会有边框

android:clickable：一定要设置成true否则没有点击效果

FloatingActionButton的位置不是根据屏幕大小来定的，必须绑定一个容器，而且 FloatingActionButton必须在CoordinatorLayout里边才生效，其它譬如LinearLayout之类的都不可以

绑定哪个框架app:layout\_anchor="@id/appbar"

在那个框架的位置app:layout\_anchorGravity="bottom|right|end"

CoordinatorLayout的组件名

android.support.design.widget.CoordinatorLayout

androidx.coordinatorlayout.widget.CoordinatorLayout

示例代码：

<androidx.coordinatorlayout.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"

tools:context=".MainActivity">

<com.google.android.material.floatingactionbutton.FloatingActionButton

android:id="@+id/fab"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="16dp"

app:layout\_anchor="@id/app\_bar"

app:layout\_anchorGravity="bottom|end"

android:onClick="aas"

app:srcCompat="@android:drawable/ic\_dialog\_email" />

## 引入公共部分

<include layout="@layout/公共"/>

## 引用网页 WebView

AndroidManifest.xml 这个很重要

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

引用远程

WebView webView=(WebView) findViewById(R.id.web\_view);  
webView.getSettings().setJavaScriptEnabled(true);

//设置WebView属性，能够执行Javascript脚本  
  
webView.setWebViewClient(new WebViewClient());  
webView.loadUrl("https://www.baidu.com");

调用本地

在main下建文件夹asset，里边放入静态网页

WebView webView = findViewById(R.id.web\_view);  
WebSettings webSettings = webView.getSettings();  
webView.getSettings().setJavaScriptEnabled(true);  
// 设置可以访问文件  
webSettings.setAllowFileAccess(true);  
webSettings.setAllowContentAccess(true);  
webSettings.setDomStorageEnabled(true);  
webView.loadUrl("file:///android\_ asset/index.html");

## touch

整个页面中监听 重写onTouchEvent方法

@Override

public boolean onTouchEvent(MotionEvent event) {

int action=event.getAction();//得到当前用户动作

float x=event.getX();

float y=event.getY();

switch (action){

case MotionEvent.ACTION\_DOWN:

Log.i("TEST","按下:x="+x+",y="+y);

break;

case MotionEvent.ACTION\_MOVE:

Log.i("TEST","移动:x="+x+",y="+y);

break;

case MotionEvent.ACTION\_UP:

Log.i("TEST","弹起:x="+x+",y="+y);

break;

}

//默认返回false,表示在自身方法中没有做完，需向上传递

//如果认为方法中事件做完了，可以返回true

return super.onTouchEvent(event);

}

对固定区域进行监听，好像只有View可以实现

<View

android:id="@+id/ss"

android:layout\_width="match\_parent"

android:layout\_height="400dp"

android:text="Hello World!"

android:background="#ff4f02"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

Activity中写一个方法内容为下

View view = findViewById(R.id.ss);

view.setOnTouchListener(new View.OnTouchListener(){

@Override

public boolean onTouch(View v, MotionEvent event) {

float x=event.getX();

float y=event.getY();

if(event.getAction()==MotionEvent.ACTION\_MOVE)

{

Log.i("TEST","移动:x="+x+",y="+y);

}

return true;

}

});

## 下拉标题栏变化AppbarLayou

结构

>CoordinatorLayout

>AppBarLayout

>CollapsingToolbarLayout

>ImageView/TextView/自定义视图

>Toolbar

>NestedScrollView

>RecyclerView/自定义视图

>FloatingActionButton

AppbarLayout是一种支持响应滚动手势的app bar布局

AppbarLayout 还是一样的，必须在CoordinatorLayout里边实现

跟AppBarLayout同级的必须有NestedScrollView

AppBarLayout子级里必须有CollapsingToolbarLayout专门来实现子布局内不同元素响应滚动细节的布局.

使用布局首先得把activty 设为无标题，详细方法参照： 标题栏ToolBar

android:theme="@style/AppTheme.NoActionBar"

【**CoordinatorLayout**】

详解

app:contentScrim="?attr/colorBackgroundFloating" 当被折叠到顶部固定大小时候的背景颜色

app:layout\_scrollFlags="scroll|exitUntilCollapsed"

scroll|enterAlways 折叠了没有标题栏

scroll|exitUntilCollapsed 折叠了标题栏还在顶端

(1)scroll: 值设为scroll的View会跟随滚动事件一起发生移动。就是当指定的ScrollView发生滚动时，该View也跟随一起滚动，就好像这个View也是属于这个ScrollView一样。

(2) enterAlways: 值设为enterAlways的View,当任何时候ScrollView往下滚动时，该View会直接往下滚动。而不用考虑ScrollView是否在滚动到最顶部还是哪里.

(3) exitUntilCollapsed：值设为exitUntilCollapsed的View，当这个View要往上逐渐“消逝”时，会一直往上滑动，直到剩下的的高度达到它的最小高度后，再响应ScrollView的内部滑动事件。

(4) enterAlwaysCollapsed：是enterAlways的附加选项，一般跟enterAlways一起使用，它是指，View在往下“出现”的时候，首先是enterAlways效果，当View的高度达到最小高度时，View就暂时不去往下滚动，直到ScrollView滑动到顶部不再滑动时，View再继续往下滑动，直到滑到View的顶部结束

(5) snap：简单理解，就是Child View滚动比例的一个吸附效果。也就是说，Child View不会存在局部显示的情况，滚动Child View的部分高度，当我们松开手指时，Child View要么向上全部滚出屏幕，要么向下全部滚进屏幕，有点类似ViewPager的左右滑动

示例

<com.google.android.material.appbar.CollapsingToolbarLayout

android:id="@+id/toolbar\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

app:contentScrim="?attr/colorBackgroundFloating"

app:layout\_scrollFlags="scroll|exitUntilCollapsed"

app:toolbarId="@+id/toolbar">

自定义的子视图 layout\_collapseMode有两种模式

“pin”：固定模式，在折叠的时候最后固定在顶端；

“parallax”：视差模式，在折叠的时候会有个视差折叠的效果

【**NestedScrollView**】写法很固定

<androidx.core.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:showIn="@layout/activity\_main">

</androidx.core.widget.NestedScrollView>

整体示例

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.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=".ScrollingActivity">  
  
 <com.google.android.material.appbar.AppBarLayout  
 android:id="@+id/app\_bar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="180dp"  
 android:fitsSystemWindows="true"  
 android:theme="@style/AppTheme">  
  
 <com.google.android.material.appbar.CollapsingToolbarLayout  
 android:id="@+id/toolbar\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 app:contentScrim="?attr/colorPrimary"  
 app:layout\_scrollFlags="scroll|exitUntilCollapsed"  
 app:toolbarId="@+id/toolbar">  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 app:layout\_collapseMode="pin"  
 app:popupTheme="@style/AppTheme" />  
  
 </com.google.android.material.appbar.CollapsingToolbarLayout>  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
>  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="@dimen/text\_margin"  
 android:text="@string/large\_text" />  
  
 </androidx.core.widget.NestedScrollView>  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp"  
 app:layout\_anchor="@id/app\_bar"  
 app:layout\_anchorGravity="bottom|end"  
 app:srcCompat="@android:drawable/ic\_dialog\_email" />  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

## 生成右上脚菜单optionsMenu

Res/menu建立.xml结尾的meun文件

<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="com.example.myapplication.ScrollingActivity">  
 <item  
 android:id="@+id/action\_settings"  
 android:orderInCategory="100"  
 android:title="@string/action\_settings"  
 app:showAsAction="never" />  
</menu>

Activity中重写onCreateOptionsMenu 和 onOptionsItemSelected 方法

public class ScrollingActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_scrolling);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.menu\_scrolling, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
 if (id == R.id.action\_settings) {  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

# intent意图

## 1.跳转内部页面（显式）

Intent intent = new Intent();

intent.setClass(MainActivity.this,Main2Activity.class);

startActivity(intent);

## 2.跳转内部页面（隐式）

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="定义的名字，后边调用" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

Intent intent=new Intent();  
intent.setAction("定义的名字”);  
startActivity(intent);

## 3.跳转网址并选择用什么浏览器打开

Intent intent = new Intent(Intent.ACTION\_DIAL);

intent.setData(Uri.parse("http:www.baidu.com"));

startActivity(intent);

## 4.打电话

Intent intent = new Intent(Intent.ACTION\_DIAL);

intent.setData(Uri.parse("tel:10086"));

startActivity(intent);

## 5.向下一个活动传参

可以传多个值，并且写法一样

Intent intent = new Intent();  
intent.setClass(MainActivity.this,lunbo.class);  
intent.putExtra("pageData","主页面传过来的数据");

intent.putExtra("pageNum",1231);  
startActivity(intent);

接收 取int 和取string

Intent intent=getIntent();  
String data= intent.getStringExtra("pageData") ;  
int data2= intent.getIntExtra("pageNum",-1); //当这个值没有取到的时候，默认值为-1

[以Bundle的形式去传参（也就是以Object），传的时候参数为putExtras]

Intent intent = new Intent();  
intent.setClass(this,MainActivity.class);  
Bundle bundle=new Bundle();  
bundle.putString("name","张三");  
bundle.putInt("age",19);  
intent.putExtras(bundle);  
startActivity(intent);

两种取法

第一种跟之前单个传的取法一样

String data1= intent.getStringExtra("name");  
int data2= intent.getIntExtra("age",-1);

第二种以Bundel去取

Bundle bundle=intent.getExtras();  
String name=bundle.getString("name");  
int age=bundle.getInt("age");

## 6.返回数据给上一个活动

使用startActivityForResult()启动一个Activity

重写onActivityResult()方法接受返回

在启动的Activity中，调用方法setResult()设置需要返回的内容(跳转过去的那个活动)

跳转过去的那个Activity，有个按钮点击事件，单独写还不行，只能监听

final Intent intent=getIntent();  
final String name= intent.getStringExtra("name");  
final int age= intent.getIntExtra("age",-1); //当这个值没有取到的时候，默认值为-1  
Button btnBack=(Button)findViewById(R.id.buttons);  
btnBack.setOnClickListener(  
 new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String newtext=name+age;  
 intent.putExtra("result",newtext);  
 setResult(0x002,intent);  
 finish(); //关闭当前Activity  
 }  
 }  
);

按钮点击事件

Intent intent = new Intent();  
intent.setClass(this,MainActivity.class);  
intent.putExtra("name","主页面传过来的数据");  
intent.putExtra("age",1231);  
startActivityForResult(intent,0x0001);

重写onActivityResult方法

/\*  
\* 当有返回值自动回调  
 \*/  
@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 Log.i("TEST",requestCode+"");  
 Log.i("TEST",requestCode+"");  
 if(requestCode==0x0001 && requestCode==0x002){ }  
 String result=data.getStringExtra("result");  
 Toast.makeText(this,result+"",Toast.LENGTH\_LONG).show();  
 super.onActivityResult(requestCode, resultCode, data);  
}

# Fragment片段

onAttach()：Fragment和Activity相关联时调用。可以通过该方法获取Activity引用，还可以通过getArguments()获取参数。

onCreate()：Fragment被创建时调用。

onCreateView()：创建Fragment的布局。

onActivityCreated()：当Activity完成onCreate()时调用。

onStart()：当Fragment可见时调用。

onResume()：当Fragment可见且可交互时调用。

onPause()：当Fragment不可交互但可见时调用。

onStop()：当Fragment不可见时调用。

onDestroyView()：当Fragment的UI从视图结构中移除时调用。

onDestroy()：销毁Fragment时调用。

onDetach()：当Fragment和Activity解除关联时调用。

上面的方法中，只有onCreateView()在重写时不用写super方法，其他都需要。

View parentAvtivity = inflater.inflate(R.layout.activity\_main,container,false);

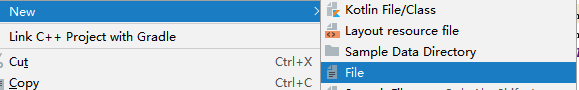
parentAvtivity.getContext()

在子Fragment中，需要用到this的地方，可以用这个替换

## 1.创建基本的Fragment

新建xml文件

对着layout点右键-new-file，其它也可以，但内容要以LineView一类的开头



<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="这是Fragment里的内容"/>  
</LinearLayout>

直接写LinearLayout会报错

加上xmlns:android="http://schemas.android.com/apk/res/android"

建类继承Fragment

对着Activity名点右键-new-java Class



package activety.fra;  
  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
public class leftfragment extends Fragment {  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.leftfrg, container, false);  
 return view;  
 }  
}

这里边的内容基本上是固定写法，继承Fragment并且修改onCreateView方法

调用

<fragment  
 android:id="@+id/l"  
 android:name="activety.fra.leftfragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
/>

id必须有，不然报错

name为Fragment的包名

## 2.Activity点击事件修改Fragment里的内容

在onViewCreated方法内定义变量，自己创个方法暴露出来，后边调用

import android.os.Bundle;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
public class leftfragment extends Fragment {  
 private TextView textView;  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.leftfrg, container, false);  
 return view;  
 }  
 @Override  
 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 textView=(TextView) view.findViewById(R.id.textA);  
 super.onViewCreated(view, savedInstanceState);  
 }  
 //自定义一个方法，后边获取到这个类直接 类.方法名调用  
 public void haha(String text)  
 {  
 this.textView.setText(text);  
 }  
}

Activity

private leftfragment skA; //自己创的fragment在这里成为实体类  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 FragmentManager mfragmentManager=getSupportFragmentManager(); //FragmentManager的引用要和Fragment来自同一个包  
 skA=(leftfragment) mfragmentManager.findFragmentById(R.id.sheep);

//skA为自己定义的fragment类 sheep为xml中fragment标签的id  
}  
public void aasd(View view){  
 skA.haha(" 什么谦恭");  
}

## 动态修改FrameLayout里的fragment页面

xml页面创建FrameLayout

<FrameLayout  
 android:id="@+id/sheep"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 android:layout\_weight="1"  
</FrameLayout>

FrameLayout与Fragment标签都是用来放Fragment的，区别是，Fragment标签只能设置固定的Fragment，而FrameLayout中可以通过FragmentManager更换Fragment

Activity页面定义方法，并在onClick和初始化中调用

replaceFragment(new otherleftfragment()); //otherleftfragment为fragment类名

private void replaceFragment(Fragment fragment)

{

FragmentManager mframentmanage2=getSupportFragmentManager();

FragmentTransaction transaction=mframentmanage2.beginTransaction();

transaction.replace(R.id.sheep,fragment);

transaction.commit();

}

## Fragment的一些技巧

findViewById报错的时候 在前边加getActivity() getActivity().findViewById

# ViewPager

Xml

<android.support.v4.view.ViewPager  
 android:id="@+id/vp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 />

定义两个变量

private ViewPager mvp;  
private List<ImageView> mlist;

onCreate方法中

setContentView(R.layout.activity\_main); //这句是用来干嘛啊  
mvp=(ViewPager) findViewById(R.id.vp);  
  
 mlist=new ArrayList<ImageView>();  
  
int[] imageRes=new int[] { R.drawable.swiper1,R.drawable.swiper2,R.drawable.swiper3,R.drawable.swiper4};  
for (int i=0;i<imageRes.length;i++)  
{  
 ImageView iv=new ImageView(this);  
 iv.setBackgroundResource(imageRes[i]);  
 mlist.add(iv);  
}  
mvp.setAdapter(new MyAdapter());

定义一个类

class MyAdapter extends PagerAdapter{  
 //用来决定ViewPager中到底可以显示多少个子视图  
 @Override  
 public int getCount() {  
 return mlist!=null?mlist.size():0 ;  
 }  
 //判断是否需要重新生成新的子视图  
 @Override  
 public boolean isViewFromObject(@NonNull View view, @NonNull Object o) {  
 return view==o;  
 }  
 //产生一个新视图  
 @NonNull  
 @Override  
 public Object instantiateItem(@NonNull ViewGroup container, int position) {  
  
 container.addView(mlist.get(position));  
 return mlist.get(position);  
 }  
 //从viewpager中移除某个条目  
 @Override  
 public void destroyItem(@NonNull ViewGroup container, int position, @NonNull Object object) {  
  
 container.removeView(mlist.get(position));  
 }  
}

# 线程及HTTP请求

## 1.线程

这个实例主要是把线程里的内容返回到界面中

定义变量并重新Hander方法

private Handler handler=new Handler(){

@Override

public void handleMessage(Message msg) {

TextView text=(TextView) findViewById(R.id.textA);

//处理消息，运行在主线程中

switch (msg.what){

case 0x0001: //获取那边传过来的标识

int index=msg.arg1; //获取那边传过来的值

text.setText(""+index); //给UI线程中的控件赋值

break;

}

}

};

在一个方法中定义一个线程并重写它的run方法

new Thread(){

@Override

public void run() {

for (int i = 0; i <100 ; i++) {

//子线程不可更新UI线程中的控件的属性

Message msg=new Message();

msg.what=0x0001;//消息的what是消息标识，必须要有

msg.arg1=i; //把i传到主线程上

handler.sendMessage(msg);

try {

Thread.sleep(1000); //过1秒执行一次

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}.start();

# 列表：ListView/GridView/RecyclerView

Adapter适配器（ListView、GridView）

分割线

android:divider="#25629f"

android:dividerHeight="1px" //设置了divider就必须设置这个

## ArrayAdapter 只可以传一个文本

[虽然只可以加一个变量进去，但一样可以加带图片的list]

ArrayAdapter的数据类型只能是Array

定义一个xml,后边直接调用它，里边要给ID，后边用得上

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:app="http://schemas.android.com/apk/res-auto"  
android:layout\_width="wrap\_content"  
android:layout\_height="wrap\_content">  
<ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 app:srcCompat="@mipmap/ic\_launcher\_round" />  
<TextView  
 android:layout\_width="wrap\_content"  
 android:id="@+id/textk"  
 android:layout\_height="wrap\_content"  
 android:text="我是一个复杂的Toast" />  
</LinearLayout>

定义变量

private String[] aa={"张土木工程","土木工程","城在","a1","a1","a1","a1","a1","a1","a1","a1","a1","a1","a1","a1","a1"};

ListView listView=(ListView) findViewById(R.id.listview);  
ArrayAdapter<String> arrayAdapter=new ArrayAdapter<String>(  
 this, //上下文  
 R.layout.toastview, //显示的布局名称  
 R.id.textk, //这个布局名称中，数据要显示的ID  
 aa //数据源  
);  
listView.setAdapter(arrayAdapter);

使用andoid自带的list,并监听

ListView listView=(ListView) findViewById(R.id.listview);  
  
 listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
 //parent为每一项的父容器，也就是那个ListView  
 // view是每一项，也就是item  
 //position是每一项的索引  
 //id 没说哪的id  
 Toast.makeText(MainActivity.this,"你点了"+aa[position],Toast.LENGTH\_LONG).show();  
  
 }  
 });  
  
  
 ArrayAdapter<String> arrayAdapter=new ArrayAdapter<String>(  
 this, //上下文  
 android.R.layout.simple\_list\_item\_1,// R.layout.toastview, //显示的布局名称  
 android.R.id.text1, //这个布局名称中，数据要显示的ID  
 aa //数据源  
 );  
 listView.setAdapter(arrayAdapter);

## SimpleAdapter多个String

可以显示比较复杂的列表，包括每行显示图片（比较难）、文字等，但不能对列表进行后期加工

SimpleAdapter的数据类型可以是第一个值是String类型的map、 json转的map

定义一个layout

<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">  
<TextView  
 android:id="@+id/img1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:srcCompat="@mipmap/ic\_launcher\_round" />  
<TextView  
 android:id="@+id/text1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="我是一个复杂的Toast" />  
<TextView  
 android:id="@+id/img2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 app:srcCompat="@mipmap/ic\_launcher\_round" />  
</LinearLayout>

private List<Map<String,String>> data;  
  
ListView listView;  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 fullData(); //加入数据  
 listView=(ListView) findViewById(R.id.listview);  
 SimpleAdapter simpleAdapter=new SimpleAdapter(  
 this, //上下文  
 data, //map型数据源  
 R.layout.toastview, //项布局  
 new String[]{"name","age","sex"}, //用String型数组显示数据源中key的名字  
 new int[] {R.id.img1,R.id.text1,R.id.img2}); //显示数据对应的控件名称  
  
 listView.setAdapter(simpleAdapter);  
}  
  
private void fullData() {  
 data=new ArrayList<Map<String, String>>();  
 Map<String,String> stu1=new HashMap<>();  
 stu1.put("name","张三");  
 stu1.put("age","20");  
 stu1.put("sex","男");  
 data.add(stu1);  
  
 Map<String,String> stu2=new HashMap<>();  
 stu2.put("name","李四");  
 stu2.put("age","22");  
 stu2.put("sex","女");  
 data.add(stu2);  
  
 Map<String,String> stu3=new HashMap<>();  
 stu3.put("name","王五");  
 stu3.put("age","222");  
 stu3.put("sex","男女");  
 data.add(stu3);  
}

## baseAdapter (常用)

在Activity中加入ListView

<ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:divider="#25629f"  
 android:dividerHeight="1px"  
 android:id="@+id/ssk"></ListView>

建布局文件list\_item.xml

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:orientation="horizontal"  
 android:layout\_height="match\_parent">  
 <ImageView  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:src="@drawable/lo\_1"  
 android:id="@+id/img"/>  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="12321"  
 android:id="@+id/title1"/>  
</LinearLayout>

import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
  
 private ListView listView;  
 private int[] imgResIds={R.drawable.lo\_1,R.drawable.lo\_2,R.drawable.lo\_3,R.drawable.lo\_4,R.drawable.lo\_5,R.drawable.lo\_6};  
 private String[] names={"Dog1","Dog2","Dog3","Dog4","Dog5","Dog6"};  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 listView=(ListView) findViewById(R.id.listview1); //绑定ListView  
 listView.setAdapter(new MyAdapter());  
 }  
 class MyAdapter extends BaseAdapter{  
 //数据源中项的个数  
 @Override  
 public int getCount() {  
 return names.length;  
 }  
 //项  
 @Override  
 public Object getItem(int position) {  
 return position;  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
 //项显示的View position 项的索引  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 //解析布局文件，成为View  
 View view;  
 ViewHolder holder=null;  
 if(convertView==null){ //使用converView来提高效率  
 LayoutInflater inflater=LayoutInflater.from(MainActivity.this);  
 view=inflater.inflate(R.layout.list\_item,null);  
 holder=new ViewHolder(view);  
 view.setTag(holder); //在view上保存程序所需要的数据  
 }  
 else {  
 view=convertView;  
 holder=(ViewHolder) view.getTag(); //取出之前的数据  
 }  
   
 //填充当前项的数据  
 //图片的写法  
 ImageView image=holder.getImage(); //这个图片ID是布局文件中显示图片的ImageView的ID  
 image.setImageResource(imgResIds[position%names.length]);  
 //文本的写法  
 TextView text=holder.getTitle(); //这个文本ID是布局文件中显示图片的TextView的ID  
 text.setText(names[position%names.length]);  
  
 return view;  
 }  
 }  
  
 //为了优化，创建持有者类  
 class ViewHolder{  
 private View view;  
 private ImageView image;  
 private TextView title;  
 public ViewHolder(View view)  
 {  
 this.view=view;  
 }  
 public ImageView getImage(){  
 if (image==null)  
 {  
 image= (ImageView) view.findViewById(R.id.img);  
 }  
 return image;  
 }  
 public TextView getTitle(){  
 if(title==null)  
 {  
 title=(TextView) view.findViewById(R.id.title1);  
 }  
 return title;  
 }  
  
 }  
}

## ListView的点击事件

对ListView点击其中任意一行的事件的监听

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
 //parent为每一项的父容器，也就是那个ListView  
 // view是每一项，也就是item  
 //position是每一项的索引  
 //id 没说哪的id  
 Toast.makeText(MainActivity.this,"你点了"+aa[position],Toast.LENGTH\_LONG).show();  
  
 }  
 });

## GridView

GridView 和ListView的适配器使用方法一模一样

在XML GridView标签中

android:numColumns="4" 这个代表一行有几列

android:verticalSpacing="20dp" 行与行之间的间距

对GridView点击其中任意一行的事件的监听

gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
 //parent为每一项的父容器，也就是那个ListView  
 // view是每一项，也就是item  
 //position是每一项的索引  
 //id 没说哪的id  
 Toast.makeText(MainActivity.this,"你点了"+data[position],Toast.LENGTH\_LONG).show();  
  
 }  
});

## 更新GridView 适配器实例

adapter.notifyDataSetChanged();

public class Main2Activity extends AppCompatActivity {  
 private Button butA;  
 private EditText editA;  
 private GridView gridA;  
 //数据源  
 List<String> allText=new ArrayList<String>();  
 //适配器  
 ArrayAdapter<String> adapter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main2);  
 initControl();  
 adapter=new ArrayAdapter<String>(  
 this,  
 android.R.layout.simple\_list\_item\_1,  
 android.R.id.text1,  
 allText  
 );  
 gridA.setAdapter(adapter);  
 }  
 private void initControl(){  
 butA=(Button)findViewById(R.id.butA);  
 editA=(EditText) findViewById(R.id.editA);  
 gridA=(GridView) findViewById(R.id.gridA);  
 }  
 public void addData(View view) {  
 String name="321321";  
 Log.i("text",name);  
 allText.add(name);  
 //适配器发出通知更新gridView  
 adapter.notifyDataSetChanged();  
  
 }  
}

## RecyclerView(常用)

build.gradle引入

v7: implementation 'com.android.support:recyclerview-v7:28.0.0-alpha1'

androidX: implementation 'androidx.recyclerview:recyclerview:1.0.0'

下边以androidX为例

<?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">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/recycler\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

</LinearLayout>

1. 引入类库
2. 写布局文件RecyclerView
3. 准备数据
4. 设置类继承RecyclerView.Adapter
5. 新建类泛型
6. 设置适配器 完了setLayoutManager

流程总结

ListView

1. 在Activity中建组件ListView
2. 建一个试图模型组件item.xml
3. 准备数据并绑定ListView
4. 适配器+持有者，继承BaseAdapter

4个方法 getCount getItem getItemId getView 在getView方法中引入持有者类

1. 设置适配器

listView.setAdapter(new MyAdapter());

1. 分割线

android:divider="#25629f"

android:dividerHeight="1px" //设置了divider就必须设置这个

1. Click监听

listView.setOnItemClickListener

RecyclerView

1. 在Activity中建组件RecyclerView
2. 建一个试图模型组件item.xml
3. 准备数据并绑定RecyclerView
4. 适配器+持有者
5. 设置适配器

MyRecyclerViewAdapter adapter = new MyRecyclerViewAdapter(MainActivity.this,datas);  
recyclerview.setAdapter(adapter);

1. 分割线

定义形状并加入 定义样式的方式好像不生效  
 DividerItemDecoration divider = new DividerItemDecoration(this,DividerItemDecoration.VERTICAL);  
 divider.setDrawable(ContextCompat.getDrawable(this,R.drawable.divider));  
 recyclerview.addItemDecoration(divider);

1. Click监听

adapter.setOnItemClickListen

# 系统服务

## 1.震动器

<uses-permission android:name="android.permission.VIBRATE"/>

Vibrator myVibrator = (Vibrator) getSystemService(Service.VIBRATOR\_SERVICE);  
myVibrator.cancel();  
myVibrator.vibrate(new long[]{0, 50}, -1);

数组的a[0]表示静止的时间，a[1]代表的是震动的时间，然后数组的a[2]表示静止的时间，a[3]代表的是震动的时间……依次类推下去

第二参数表示从哪里开始循环，比如这里的0表示这个数组在第一次循环完之后会从下标0开始循环到最后，这里的如果是-1表示不循环。

## 2.调节音量

AudioManager audioManager=(AudioManager) getSystemService(Context.AUDIO\_SERVICE);  
/\*  
\* 声音类型：手机、闹钟、音乐……  
\* 调大、调小  
\* 显示界面、震动……  
 \*/  
audioManager.adjustStreamVolume(AudioManager.STREAM\_ALARM,AudioManager.ADJUST\_RAISE,AudioManager.FLAG\_SHOW\_UI);  
  
/\*  
 \* 声音类型：手机、闹钟、音乐……  
 \* 音量的值，1-7  
 \* 显示界面、震动……  
 \*/  
audioManager.setStreamVolume(AudioManager.STREAM\_ALARM,1,AudioManager.FLAG\_SHOW\_UI);

3.定时服务  
写一个服务

public class myService extends Service {  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 Toast.makeText(getApplicationContext(),"时间到了",Toast.LENGTH\_LONG).show();  
 return super.onStartCommand(intent, flags, startId);  
 }  
}

注册

<service android:name=".myService">  
 <intent-filter>  
 <action android:name="ss"/>  
 </intent-filter>  
</service>

Activity实现

AlarmManager alarmManager=(AlarmManager) getSystemService(ALARM\_SERVICE);  
 /\*  
 \* 参数一：type  
 \* ELAPSED\_REALTIME 睡眠时候不可用，使用相对时间  
 \* RTC 睡眠时候不可用，使用绝对时间  
 \* ELAPSED\_REALTIME\_WAKEUP 睡眠时唤醒系统  
 \* RTC\_WAKEUP 使用绝对时间，睡眠时唤醒系统  
 \* 参数二：triggerAtMillis 触发时间  
 \* 参数三：operation  
 \* 用到的是静态方法 可以启动Activity、启动服务、发送广播  
 \*/  
 long time=System.currentTimeMillis();  
 time+=1000;  
 Intent intent = new Intent("ss");  
 intent.setPackage(this.getPackageName()); //兼容Android 5.0  
 this.startService(intent);  
 PendingIntent pi=PendingIntent.getService(this,0,intent,0);  
alarmManager.set(AlarmManager.RTC\_WAKEUP,time,pi);

## 3.位置服务

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

import android.Manifest;  
import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.PackageManager;  
import android.location.Criteria;  
import android.location.Location;  
import android.location.LocationListener;  
import android.location.LocationManager;  
import android.media.AudioManager;  
import android.support.v4.app.ActivityCompat;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
  
public class MainActivity extends AppCompatActivity {  
 private static final String TAG = "LocationService";  
 private SharedPreferences sp;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 }  
 public void voice(View view) {  
 LocationManager locationManager = (LocationManager) getSystemService(LOCATION\_SERVICE);  
 LocationManager ls = (LocationManager) getSystemService(LOCATION\_SERVICE);  
 Criteria criteria = new Criteria();  
 criteria.setAccuracy(Criteria.ACCURACY\_FINE);  
 String provider = ls.getBestProvider(criteria, true);  
 if (provider == null) {  
 Log.d(TAG, "provider equals null");  
 return;  
 }  
 MyListener listener = new MyListener();  
 if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  
 return;  
 }  
 ls.requestLocationUpdates(provider, 0, 0, listener);  
 Location location = ls.getLastKnownLocation(provider);  
 if (location == null){  
 Log.d(TAG, "location equals null");  
 return;  
 }  
 sp = getSharedPreferences("config", Context.MODE\_PRIVATE);  
 saveLocation(location);  
 Log.d(TAG, "locationService");  
 }  
  
 public void saveLocation(Location location){  
 double longitude = location.getLongitude();  
 double altitude = location.getLatitude();  
 float accuracy = location.getAccuracy();  
 String pos = "j:" + longitude + "\tw:" + altitude  
 + "\ta:" + accuracy + "\n";  
 SharedPreferences.Editor edit = sp.edit();  
 edit.putString("location", pos);  
 edit.commit();  
 Log.d(TAG, pos);  
 }  
  
 private class MyListener implements LocationListener{  
 @Override  
 public void onLocationChanged(Location location) {  
 saveLocation(location);  
 Log.d("LocationService", "onLocationChanged");  
 }  
 @Override  
 public void onStatusChanged(String provider, int status, Bundle extras) {  
 }  
  
 @Override  
 public void onProviderEnabled(String provider) {  
 }  
 @Override  
 public void onProviderDisabled(String provider) {  
 }  
 }  
}

## 4.调用相机

Intent intent = new Intent("android.media.action.IMAGE\_CAPTURE");  
intent.putExtra(MediaStore.EXTRA\_OUTPUT,111);  
startActivityForResult(intent,1);

# 开发小技巧

## 解决启动白屏的问题

在主样式里边加上这句话

<item name="android:windowIsTranslucent">true</item>

# 绘制与动画

## 最基本的绘制示例

新建类

package com.yargs.draw;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.util.AttributeSet;

import android.view.View;

import androidx.annotation.Nullable;

public class iCicle extends View {

Paint paint;

public iCicle(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

}

// 重写draw方法

@Override

public void draw(Canvas canvas) {

super.draw(canvas);

// 实例化画笔对象

paint = new Paint();

// 给画笔设置颜色

paint.setColor(Color.RED);

// 设置画笔属性

paint.setStyle(Paint.Style.FILL);//画笔属性是实心圆

// paint.setStyle(Paint.Style.STROKE);//画笔属性是空心圆

paint.setStrokeWidth(8);//设置画笔粗细

/\*四个参数：

参数一：圆心的x坐标

参数二：圆心的y坐标

参数三：圆的半径

参数四：定义好的画笔

\*/

canvas.drawCircle(getWidth() / 2, getHeight() / 2, 200, paint);

}

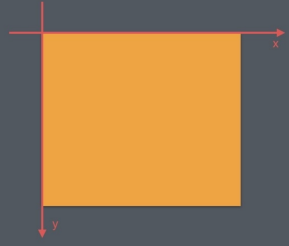
}

在视图中引用

<com.yargs.draw.iCicle

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>



View 的坐标 (x, y) 处，指的就是相对它的左上角那个点的水平方向 x 像

素、竖直方向 y 像素的点。例如，(300, 300) 指的就是左上角的点向右 300 、向下

300 的位置； (100, -50) 指的就是左上角的点向右 100 、向上 50 的位置。

## 各种图形线条的绘制

drawCircle 圆形

drawRect 正方形

drawPath 三角形

drawBitmap 图片

drawText 文字

package yargs.draw;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Path;  
import android.util.AttributeSet;  
import android.util.Log;  
import android.view.View;  
  
import androidx.annotation.Nullable;  
  
public class iCirecle extends View {  
 public iCirecle(Context context, @Nullable AttributeSet attrs) {  
 super(context, attrs);  
 }  
 Paint paint = new Paint();  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 paint.setColor(0x99880000); // 设置颜色  
 paint.setStyle(Paint.Style.FILL); // FILL 实心 STROKE空心 FILL\_AND\_STROKE两种模式一起  
 paint.setStrokeWidth(0); // 设置线条宽度或点的宽度 上一个模式设为空心或者FILL\_AND\_STROKE才有用  
 paint.setTextSize(18); // 设置文字大小  
 paint.setAntiAlias(true); // 设置抗锯齿开关 这个几乎可以说默认就得开着，不然图形很难看  
 canvas.drawColor(0x00ffffff); // 设置前景色  
 // 绘制一个圆  
 Log.e("onDraw: ",getWidth()+"");  
 Log.e("onDraw: ",getHeight()+"");  
 canvas.drawCircle(getWidth()-100, 0, 200, paint);  
 canvas.drawCircle(getWidth()-200, 200, 50, paint);  
 // 绘制一个正方形  
 // 这里上下左右的意思是每个点在屏幕的位置，相减得到的才是正方形的宽  
 canvas.drawRect(getWidth()-200, 250, getWidth()-10, 500, paint);  
 // 绘制一个点  
 paint.setStrokeWidth(50); // 点的大小  
 paint.setStrokeCap(Paint.Cap.ROUND); // 圆头ROUND 平头BUTT 方头SQUARE  
 canvas.drawPoint(getWidth()-100, 550,paint);  
  
 // 绘制一群点  
 float[] pointsA = {0, 0, getWidth()-200, 600,getWidth()-50, 600, getWidth()-200, 700, getWidth()-50, 700}; // 两个是一个点 绘制四个点：(50, 50) (50, 100) (100, 50) (100, 100)  
 canvas.drawPoints(pointsA, 2 /\* 跳过两个数，即前两个 0 \*/, 8 /\* 一共绘制 8 个数（4 个点）\*/, paint);  
  
 // 绘制一个椭圆 只能画横着的或竖着的  
 canvas.drawOval(getWidth()-200, 800, getWidth()-50, 900, paint);  
  
 // 绘制一条线  
 paint.setStrokeWidth(10);  
 canvas.drawLine(getWidth()-200, 1000, getWidth()-50, 1050, paint);  
  
 // 绘制一条折线  
 //这的折线不是每个点连起来，这里还是两组两组画一条线  
 float[] pointsB = {getWidth()-200, 1100, getWidth()-50, 1100, getWidth()-50, 1100, getWidth()-50, 1200,getWidth()-50, 1200, getWidth()-200, 1150, getWidth()-50, 1150};  
 canvas.drawLines(pointsB, paint);  
  
 // 绘制一个圆角矩形 left , top , right , bottom 是四条边的坐标，rx 和 ry 是圆角的横向半径和纵向 半径。  
 canvas.drawRoundRect(50, 500, 250, 600, 50, 50, paint);  
  
 // 绘制扇  
 canvas.drawArc(50, 700, 250, 900, -110, 110, true, paint);  
 paint.setColor(0x99ff4f02); // 设置颜色  
 canvas.drawArc(50, 700, 250, 900, 0, 90, true, paint);  
 paint.setColor(0x99ff0000); // 设置颜色  
 canvas.drawArc(50, 700, 250, 900, 90, 160, true, paint);  
  
 // 绘制弧形  
 paint.setColor(0x99880000); // 设置颜色  
 canvas.drawArc(50, 800, 250, 1050, 20, 140, false, paint);  
 paint.setStyle(Paint.Style.STROKE); // 画线模式  
 canvas.drawArc(50, 800, 250, 1050, 140, 80, false, paint);  
  
  
  
 //绘制自定义形状 心形  
 paint.setStyle(Paint.Style.FILL); // 画线模式  
 Path pathA = new Path();  
 pathA.addArc(200, 200, 400, 400, -225, 225);  
 pathA.arcTo(400, 200, 600, 400, -180, 225, false);  
 pathA.lineTo(400, 542);  
 canvas.drawPath(pathA,paint);  
  
 //绘制自文字  
 paint.setTextSize(50);  
 canvas.drawText("大家好，我是张三", 50, 400, paint);  
  
 //绘制直线 且闭合  
 Path pathB = new Path();  
 pathB.moveTo(50, 1300); // 由于这种绘制方法，下一个点永远是以上一个点为相对中心，moveTo,虽然是移动起点，也相当于是移动整个形状  
 pathB.lineTo(200, 1200); // 由当前位置 (0, 0) 向 (100, 100) 画一条直线  
 pathB.rLineTo(-200, 0); // 由当前位置 (100, 100) 向正右方 100 像素的位置画一  
 pathB.close();  
 canvas.drawPath(pathB,paint);  
  
 //绘制贝塞尔曲线 且闭合  
 paint.setStyle(Paint.Style.STROKE); // 画线模式  
 Path pathC = new Path();  
 pathC.moveTo(100, 500);  
 pathC.quadTo(300, 100, 600, 500);  
 pathC.rQuadTo(350, 350,100,100); // 由当前位置 (100, 100) 向正右方 100 像素的位置画一  
// pathC.close();  
 canvas.drawPath(pathC,paint);  
  
 }  
}

## 基础动画

【平移】

Animation translateAnimation = new TranslateAnimation(0,500,0,500);  
// 创建平移动画的对象：平移动画对应的Animation子类为TranslateAnimation  
// 参数分别是：  
// 1. fromXDelta ：视图在水平方向x 移动的起始值  
// 2. toXDelta ：视图在水平方向x 移动的结束值  
// 3. fromYDelta ：视图在竖直方向y 移动的起始值  
// 4. toYDelta：视图在竖直方向y 移动的结束值  
translateAnimation.setDuration(3000);  
translateAnimation.setFillAfter(true); // 留在最终的位置  
// 播放动画直接 startAnimation(translateAnimation)  
//如：  
img.startAnimation(translateAnimation);

【缩放】

Animation rotateAnimation = new ScaleAnimation(0,2,0,2,Animation.RELATIVE\_TO\_SELF,0.5f,Animation.RELATIVE\_TO\_SELF,0.5f);  
// 创建缩放动画的对象 & 设置动画效果：缩放动画对应的Animation子类为RotateAnimation  
// 参数说明:  
// 1. fromX ：动画在水平方向X的结束缩放倍数  
// 2. toX ：动画在水平方向X的结束缩放倍数  
// 3. fromY ：动画开始前在竖直方向Y的起始缩放倍数  
// 4. toY：动画在竖直方向Y的结束缩放倍数  
// 5. pivotXType:缩放轴点的x坐标的模式  
// 6. pivotXValue:缩放轴点x坐标的相对值  
// 7. pivotYType:缩放轴点的y坐标的模式  
// 8. pivotYValue:缩放轴点y坐标的相对值  
// pivotXType = Animation.ABSOLUTE:缩放轴点的x坐标 = View左上角的原点 在x方向 加上 pivotXValue数值的点(y方向同理)  
// pivotXType = Animation.RELATIVE\_TO\_SELF:缩放轴点的x坐标 = View左上角的原点 在x方向 加上 自身宽度乘上pivotXValue数值的值(y方向同理)  
// pivotXType = Animation.RELATIVE\_TO\_PARENT:缩放轴点的x坐标 = View左上角的原点 在x方向 加上 父控件宽度乘上pivotXValue数值的值 (y方向同理)  
rotateAnimation.setDuration(3000);  
// 固定属性的设置都是在其属性前加“set”，如setDuration（）  
img.startAnimation(rotateAnimation);

【旋转】

Animation rotateAnimation = new RotateAnimation(0,270,Animation.RELATIVE\_TO\_SELF,0.5f,Animation.RELATIVE\_TO\_SELF,0.5f);  
// 创建旋转动画的对象 & 设置动画效果：旋转动画对应的Animation子类为RotateAnimation  
// 参数说明:  
// 1. fromDegrees ：动画开始时 视图的旋转角度(正数 = 顺时针，负数 = 逆时针)  
// 2. toDegrees ：动画结束时 视图的旋转角度(正数 = 顺时针，负数 = 逆时针)  
// 3. pivotXType：旋转轴点的x坐标的模式  
// 4. pivotXValue：旋转轴点x坐标的相对值  
// 5. pivotYType：旋转轴点的y坐标的模式  
// 6. pivotYValue：旋转轴点y坐标的相对值  
  
// pivotXType = Animation.ABSOLUTE:旋转轴点的x坐标 = View左上角的原点 在x方向 加上 pivotXValue数值的点(y方向同理)  
// pivotXType = Animation.RELATIVE\_TO\_SELF:旋转轴点的x坐标 = View左上角的原点 在x方向 加上 自身宽度乘上pivotXValue数值的值(y方向同理)  
// pivotXType = Animation.RELATIVE\_TO\_PARENT:旋转轴点的x坐标 = View左上角的原点 在x方向 加上 父控件宽度乘上pivotXValue数值的值 (y方向同理)  
  
  
rotateAnimation.setDuration(3000);  
// 固定属性的设置都是在其属性前加“set”，如setDuration（）  
  
img.startAnimation(rotateAnimation);

【不透明度】

Animation alphaAnimation = new AlphaAnimation(1,0);  
// 步骤2：创建透明度动画的对象 & 设置动画效果：透明度动画对应的Animation子类为AlphaAnimation  
// 参数说明:  
// 1. fromAlpha:动画开始时视图的透明度(取值范围: -1 ~ 1)  
// 2. toAlpha:动画结束时视图的透明度(取值范围: -1 ~ 1)  
alphaAnimation.setDuration(3000);  
// 固定属性的设置都是在其属性前加“set”，如setDuration（）  
img.startAnimation(alphaAnimation);

## 属性动画

属性动画分为ValueAnimator和ObjectAnimator

【不透明度】

ObjectAnimator alphaAnim = ObjectAnimator.ofFloat(img, "alpha", 1.0f, 0.0f);  
//执行事件  
alphaAnim.setDuration(1000);  
//延迟  
alphaAnim.setStartDelay(300);  
alphaAnim.start();

## 组合动画AnimationSet

AnimationSet setAnimation = new AnimationSet(true);  
// 步骤1:创建组合动画对象(设置为true)

// 步骤2:定义各种动画特效

……

// 步骤3:将创建的子动画添加到组合动画里  
setAnimation.addAnimation(alpha);  
setAnimation.addAnimation(rotate);  
setAnimation.addAnimation(translate);  
setAnimation.addAnimation(scale1);  
  
img.startAnimation(setAnimation);

Android中提供的Interpolator主要有九个：

1. AccelerateDecelerateInterpolator：先加速再减速。   
   2）AccelerateInterpolator：一直加速。   
   3）AnticipateInterpolator：先往后一下，再嗖的一声一往无前。   
   4）AnticipateOvershootInterpolator：先往后一下，再一直往前超过终点，再往回收一下。   
   5）BounceInterpolator：最后像个小球弹几下。   
   6）CycleInterpolator：重复几次，感觉就是环形进度条那种，具体我还没试过。   
   7）DecelerateInterpolator：一直减速。   
   8）LinearInterpolator：线性，这个就是我们上面讲到的很均匀的了。   
   9）OvershootInterpolator：到了终点之后，超过一点，再往回走。有个参数可以定义，超过的力度。

使用示例

AnimatorSet set = new AnimatorSet();  
set.setInterpolator(new BounceInterpolator()); // 像小球一样弹几下

[视图动画（View 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/view-animation" \t "_blank)

[帧动画（Frame 动画、Drawable 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/drawable-animation" \t "_blank)

[属性动画](https://github.com/OCNYang/Android-Animation-Set/tree/master/property-animation" \t "_blank)

[触摸反馈动画（Ripple Effect）](https://github.com/OCNYang/Android-Animation-Set/tree/master/ripple-animation" \t "_blank)

[揭露动画（Reveal Effect）](https://github.com/OCNYang/Android-Animation-Set/tree/master/reveal-animation" \t "_blank)

[转场动画 & 共享元素（Activity 切换动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/transition-animation" \t "_blank)

[视图状态动画（Animate View State Changes）](https://github.com/OCNYang/Android-Animation-Set/tree/master/state-animation" \t "_blank)

[矢量图动画（Vector 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/vector-animation" \t "_blank)

[约束布局实现的关键帧动画（ConstraintSet 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/constraint-animation" \t "_blank)

## 动画的类别总结

[视图动画（View 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/view-animation" \t "_blank)

[帧动画（Frame 动画、Drawable 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/drawable-animation" \t "_blank)

[属性动画](https://github.com/OCNYang/Android-Animation-Set/tree/master/property-animation" \t "_blank)

[触摸反馈动画（Ripple Effect）](https://github.com/OCNYang/Android-Animation-Set/tree/master/ripple-animation" \t "_blank)

使用了Material主题后，波纹动画会自动应用在所有的控件上，正常情况都会去写，当然，也可以自定义形状

默认的可以使用下边一种，想要风格不同用第二种

android:background="?android:attr/selectableItemBackground" 波纹有边界

android:background="?android:attr/selectableItemBackgroundBorderless" 波纹超出边界

[揭露动画（Reveal Effect）](https://github.com/OCNYang/Android-Animation-Set/tree/master/reveal-animation" \t "_blank)

简单来说是显隐动画，而且这个动画只针对view，而且还只能是圆形放大缩小

[转场动画 & 共享元素（Activity 切换动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/transition-animation" \t "_blank)

[视图状态动画（Animate View State Changes）](https://github.com/OCNYang/Android-Animation-Set/tree/master/state-animation" \t "_blank)

[矢量图动画（Vector 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/vector-animation" \t "_blank)

[约束布局实现的关键帧动画（ConstraintSet 动画）](https://github.com/OCNYang/Android-Animation-Set/tree/master/constraint-animation" \t "_blank)

## 绘制圆形扩散特效果（实例）

新建attrs文件 values-attrs.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <declare-styleable name="SpreadView">  
 <!--中心圆颜色-->  
 <attr name="spread\_center\_color" format="color" />  
 <!--中心圆半径-->  
 <attr name="spread\_radius" format="integer" />  
 <!--扩散圆颜色-->  
 <attr name="spread\_spread\_color" format="color" />  
 <!--扩散间距-->  
 <attr name="spread\_distance" format="integer" />  
 <!--扩散最大半径-->  
 <attr name="spread\_max\_radius" format="integer" />  
 <!--扩散延迟间隔-->  
 <attr name="spread\_delay\_milliseconds" format="integer" />  
 </declare-styleable>  
</resources>

Java 文件中粘贴类 SpreadView

package com.example.amap;  
  
import android.content.Context;  
import android.content.res.TypedArray;  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.util.AttributeSet;  
import android.view.View;  
  
import androidx.annotation.Nullable;  
import androidx.core.content.ContextCompat;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* Created by louliang on 2018/10/29.  
 \*/  
  
public class SpreadView extends View{  
 private Paint centerPaint; //中心圆paint  
 private int radius = 50; //中心圆半径  
 private Paint spreadPaint; //扩散圆paint  
 private float centerX;//圆心x  
 private float centerY;//圆心y  
 private int distance = 5; //每次圆递增间距  
 private int maxRadius = 40; //最大圆半径  
 private int delayMilliseconds = 50;//扩散延迟间隔，越大扩散越慢  
 private List<Integer> spreadRadius = new ArrayList<>();//扩散圆层级数，元素为扩散的距离  
 private List<Integer> alphas = new ArrayList<>();//对应每层圆的透明度  
  
 public SpreadView(Context context) {  
 this(context,null,0);  
 }  
  
 public SpreadView(Context context, @Nullable AttributeSet attrs) {  
 this(context, attrs,0);  
 }  
  
 public SpreadView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {  
 super(context, attrs, defStyleAttr);  
 TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SpreadView, defStyleAttr, 0);  
 radius = a.getInt(R.styleable.SpreadView\_spread\_radius, radius);  
 maxRadius = a.getInt(R.styleable.SpreadView\_spread\_max\_radius, maxRadius);  
 int centerColor = a.getColor(R.styleable.SpreadView\_spread\_center\_color, ContextCompat.getColor(context, R.color.colorAccent));  
 int spreadColor = a.getColor(R.styleable.SpreadView\_spread\_spread\_color, ContextCompat.getColor(context, R.color.colorAccent));  
 distance = a.getInt(R.styleable.SpreadView\_spread\_distance, distance);  
 a.recycle();  
 centerPaint = new Paint();  
 centerPaint.setColor(centerColor);  
 centerPaint.setAntiAlias(true);  
 //最开始不透明且扩散距离为0  
 alphas.add(255);  
 spreadRadius.add(0);  
 spreadPaint = new Paint();  
 spreadPaint.setAntiAlias(true);  
 spreadPaint.setAlpha(255);  
 spreadPaint.setColor(spreadColor);  
 }  
  
 @Override  
 protected void onSizeChanged(int w, int h, int oldw, int oldh) {  
 super.onSizeChanged(w, h, oldw, oldh);  
 //圆心位置  
 centerX = w / 2;  
 centerY = h / 2;  
 }  
  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
 for (int i = 0; i < spreadRadius.size(); i++) {  
 int alpha = alphas.get(i);  
 spreadPaint.setAlpha(alpha);  
 int width = spreadRadius.get(i);  
 //绘制扩散的圆  
 canvas.drawCircle(centerX, centerY, radius + width, spreadPaint);  
 //每次扩散圆半径递增，圆透明度递减  
 if (alpha > 0 && width < 600) {  
 alpha = alpha - distance > 0 ? alpha - distance : 1;  
 alphas.set(i, alpha);  
 spreadRadius.set(i, width + distance);  
 }  
 }  
 //当最外层扩散圆半径达到最大半径时添加新扩散圆  
 if (spreadRadius.get(spreadRadius.size() - 1) > maxRadius) {  
 spreadRadius.add(0);  
 alphas.add(255);  
 }  
// //超过8个扩散圆，删除最先绘制的圆，即最外层的圆  
// if (spreadRadius.size() >= 8) {  
// alphas.remove(0);  
// spreadRadius.remove(0);  
// }  
 //中间的圆  
 canvas.drawCircle(centerX, centerY, radius, centerPaint);  
 //延迟更新，达到扩散视觉差效果  
 postInvalidateDelayed(delayMilliseconds);  
 }  
}

Layout中使用，标签名根据实际的java类地址

<com.example.amap.SpreadView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:spread\_center\_color="#ff4f02"  
 app:spread\_delay\_milliseconds="35"  
 app:spread\_distance="5"  
 app:spread\_max\_radius="90"  
 app:spread\_radius="150"  
 app:spread\_spread\_color="#ccc" />

## 揭露动画Revealeffect

【Layout】

创建一个View标签，赋予ID，可视状态visibility设为不可见

<View  
 android:id="@+id/view\_puppet"  
 android:background="#ff4f02"  
 android:layout\_width="match\_parent"  
 android:layout\_height="305dp"  
 android:visibility="gone"  
 tools:ignore="MissingConstraints" />

【java】

//外层定义变量

boolean flag = false; // 用来控制当前是显示还是隐藏状态  
private View iview; //绑定view

//onCreate 方法中绑定view

iview = findViewById(R.id.view\_puppet);

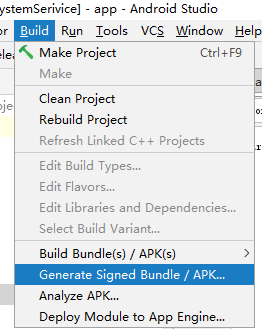
//设计点击事件之类的，最后执行该方法

public void doRevealAnimation(View view){  
 int centerX =600;  
 int centerY = 600;  
 int height = iview.getHeight();  
 int width = iview.getWidth();  
 int maxRradius = (int) Math.hypot(height, width);  
 if (height == 0) maxRradius = 1344;  
 Log.d("第一次这个值会为空", maxRradius+"");  
 if (flag) {  
 Animator animator = ViewAnimationUtils.createCircularReveal(iview, centerX, centerY, maxRradius, 0);  
 animator.setDuration(1000);  
 animator.addListener(new AnimatorListenerAdapter() {  
 @Override  
 public void onAnimationEnd(Animator animation) {  
 super.onAnimationEnd(animation);  
 iview.setVisibility(View.GONE); // 不可见 重新layout，不再占用空间 INVISIBLE 不重新layout  
 }  
 });  
 animator.start();  
 flag = false;  
 } else {  
 Animator animator = ViewAnimationUtils.createCircularReveal(iview, centerX, centerY, 0, maxRradius);  
 animator.setDuration(1000);  
 animator.addListener(new Animator.AnimatorListener() {  
 @Override  
 public void onAnimationStart(Animator animation) {  
 iview.setVisibility(View.VISIBLE);  
 }  
 @Override  
 public void onAnimationEnd(Animator animation) {  
 }  
 @Override  
 public void onAnimationCancel(Animator animation) {  
 }  
 @Override  
 public void onAnimationRepeat(Animator animation) {  
 }  
 });  
 animator.start();  
 flag = true;  
 }  
}

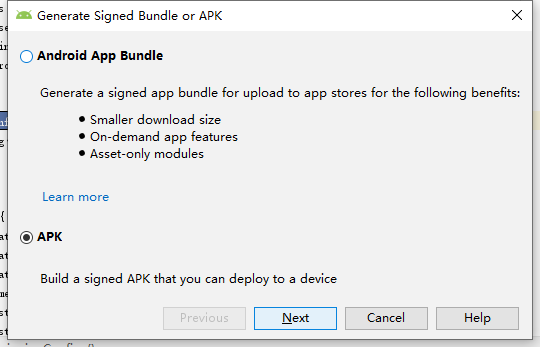
# 进阶功能实例

## 1.打包正式版本APK

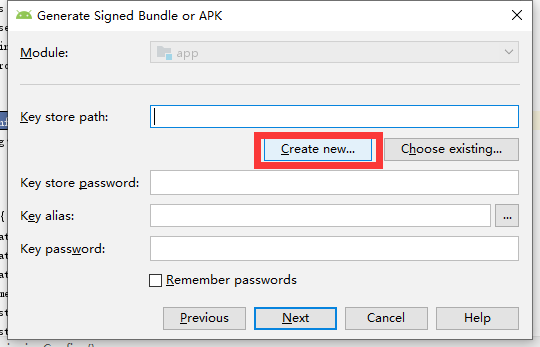
Android studio - Build - Generate Signed Bundle /APK

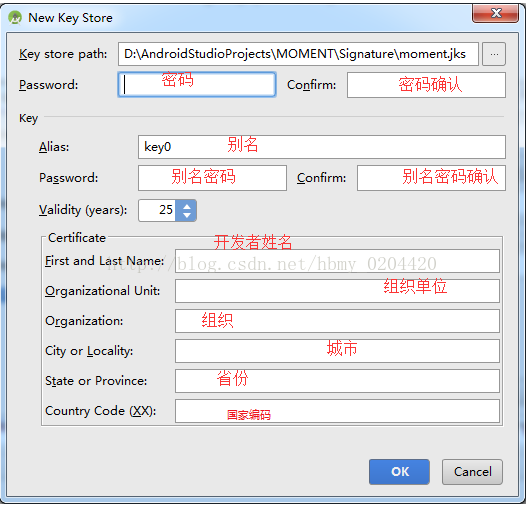


选下边第二项APK

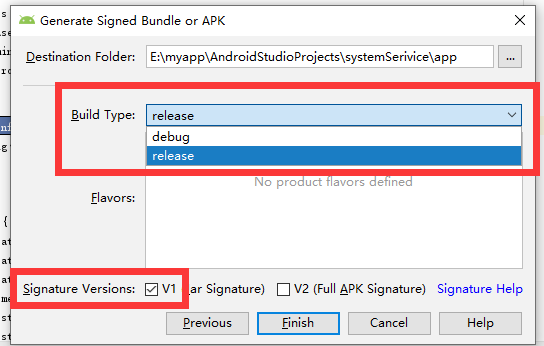


如果有生成的jks直接输入两个密码和Key Alias





选择打包正式还是测试版，下边是签名

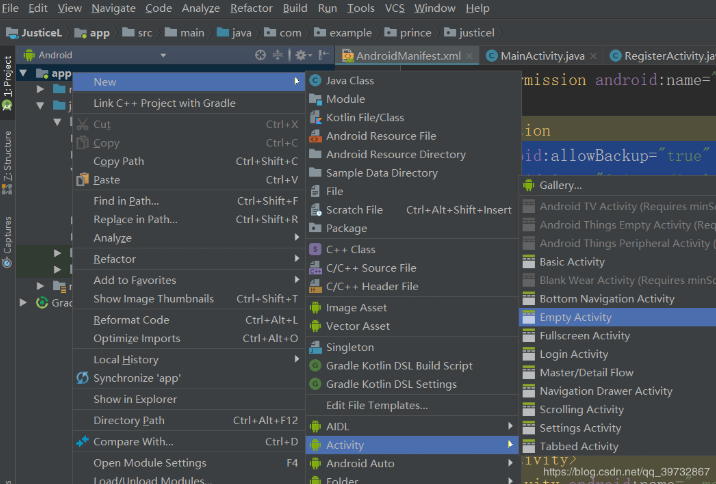


获取SHA1的java方法

public static String SHA1(Context context){  
 try {  
 PackageInfo info = context.getPackageManager().getPackageInfo(  
 context.getPackageName(), PackageManager.GET\_SIGNATURES);  
 byte[] cert = info.signatures[0].toByteArray();  
 MessageDigest md = MessageDigest.getInstance("SHA1");  
 byte[] publicKey = md.digest(cert);  
 StringBuffer hexString = new StringBuffer();  
 for (int i = 0; i < publicKey.length; i++) {  
 String appendString = Integer.toHexString(0xFF & publicKey[i])  
 .toUpperCase(Locale.US);  
 if (appendString.length() == 1)  
 hexString.append("0");  
 hexString.append(appendString);  
 hexString.append(":");  
 }  
 String result = hexString.toString();  
 return result.substring(0, result.length()-1);  
 } catch (PackageManager.NameNotFoundException e) {  
 e.printStackTrace();  
 } catch (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
 return null;  
}

## 使用启动页

【创建一个Empty Activity】



【这个Activity的控制代码】

public class SplashActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN);//隐藏状态栏  
 getSupportActionBar().hide();//隐藏标题栏  
 setContentView(R.layout.activity\_splash);  
 Thread myThread=new Thread(){//创建子线程  
 @Override  
 public void run() {  
 try{  
 sleep(5000);//使程序休眠五秒  
 Intent it=new Intent(getApplicationContext(),MainActivity.class);//启动MainActivity  
 startActivity(it);  
 finish();//关闭当前活动  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 };  
 myThread.start();//启动线程  
 }  
}

【这个 Activity 的layout显示一个图片或者其他】

<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:paddingLeft="0dp"  
android:paddingRight="0dp"  
android:paddingTop="0dp"  
android:paddingBottom="0dp"  
android:background="@drawable/splash"  
tools:context=".SplashActivity">  
</RelativeLayout>

要图片全屏 RelativeLayout 比较容易做到

【AndroidManifest.xml】

<intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>

把属于MainActivity的默认启动，剪切去刚才创建的Activity

## 解决启动时的白屏

<item name="android:windowIsTranslucent">true</item>

<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>

<!--将Activity的Theme设置成透明-->

<item name="android:windowIsTranslucent">true</item>

</style>

## 主题切换

【分别定义两组style】

<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:windowBackground">#f0f0f0</item>  
 <!--自定义属性-->  
 <item name="myBgColor">#f0f</item>  
 <item name="myButtonHeight">200dp</item>  
</style>  
  
<style name="MyTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
 <item name="colorPrimary">#000</item>  
 <item name="colorPrimaryDark">#000</item>  
 <item name="colorAccent">#ccc</item>  
 <item name="actionBarSize">80dp</item>  
 <item name="colorButtonNormal">#ccc</item>  
 <item name="android:windowBackground">#cccccc</item>  
 <!--自定义属性-->  
 <item name="myBgColor">#ff0</item>  
 <item name="myButtonHeight">300dp</item>  
</style>

【自定义属性的定义与使用】

新建attr.xml文件

<resources>  
 <attr name="myBgColor" format="color" />  
 <attr name="myButtonHeight" format="dimension" />  
</resources>

layout中的使用

android:background="?attr/myBgColor"

android:layout\_height="?attr/myButtonHeight"

【java代码】

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // 判断要写在setContentView前  
 if (useTheme) {  
 this.setTheme(R.style.MyTheme);  
 } else {  
 this.setTheme(R.style.AppTheme);  
 }  
 setContentView(R.layout.activity\_scrolling);  
  
 Button btn\_2 = (Button) findViewById(R.id.btn\_2);  
 btn\_2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 useTheme = !useTheme;  
 Intent intent = getIntent();  
 overridePendingTransition(0, 0);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_NO\_ANIMATION);  
 finish();  
 overridePendingTransition(0, 0);  
 startActivity(intent);  
 }  
 });  
}

## 使用底部导航

【添加底部目录】

添加引用

App - build.gradle

implementation 'com.android.support:design:28.0.0'

activity\_main.xml

<android.support.design.widget.BottomNavigationView

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_gravity="bottom"

app:labelVisibilityMode="labeled"

app:itemIconTint="@drawable/nav\_choose"

app:itemTextColor="@drawable/nav\_choose"

android:id="@+id/bv"

android:background="#e4e4e4"

app:menu="@menu/menu" />

这里边ID后边用得到 menu的配置参考前文，图片可以使用选中是一个样式，未选中是一个样子的那类

app:labelVisibilityMode="labeled" 在升级API28之加这句话就可以显示文字

这两句分别是图标颜色和文字颜色，完了又在里边设置选中和未选中状态不同

app:itemIconTint="@drawable/nav\_choose"

app:itemTextColor="@drawable/nav\_choose"

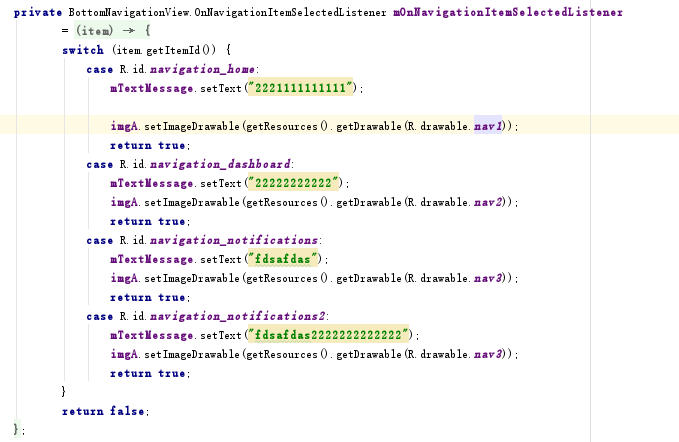
【目录切换内容切换】

在onCreate方法里

BottomNavigationView navigation = (BottomNavigationView) findViewById(R.id.bv); //bv为ID

navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener); //括号里是方法名

复制出来方法自己发生了变化，手码或者复制



switch (item.getItemId()) {  
 case R.id.navigation\_home:  
 mTextMessage.setText("2221111111111");  
  
 imgA.setImageDrawable(getResources().getDrawable(R.drawable.nav1));  
 return true;  
 case R.id.navigation\_dashboard:  
 mTextMessage.setText("22222222222");  
 imgA.setImageDrawable(getResources().getDrawable(R.drawable.nav2));  
 return true;  
 case R.id.navigation\_notifications:  
 mTextMessage.setText("fdsafdas");  
 imgA.setImageDrawable(getResources().getDrawable(R.drawable.nav3));  
 return true;  
 case R.id.navigation\_notifications2:  
 mTextMessage.setText("fdsafdas2222222222222");  
 imgA.setImageDrawable(getResources().getDrawable(R.drawable.nav3));  
 return true;

## 使用侧滑菜单DrawerLayout

>DrawerLayout

>Toolbar里的ImageView/任意控件 >加点击事件

>LinearLayout/任意控件

首先要去掉标题栏，不然标题栏目会压住侧滑菜单，重绘方式参照前文 重绘标题栏ToolBar 最好采用 样式隐藏，代码绑定 模式

>LinearLayout/任意控件 中设置 android:layout\_gravity="right" 就可以实现从右边滑出来

Activity里 可以初始化隐藏标题栏（代码参照前文），加点击打开侧栏事件

public void OpenDrewerLayout(View view){

//第一个获取DrawerLayout

//第二个不一定是LinearLayout类型，xml定义的什么类型就什么类型

DrawerLayout mDrawerLayout=(DrawerLayout) findViewById(R.id.DraLay);

LinearLayout rlRight=(LinearLayout) findViewById(R.id.rights);

if (!mDrawerLayout.isDrawerOpen(rlRight)) {

mDrawerLayout.openDrawer(rlRight);

}

}

mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_UNLOCKED); //开滑动的DrawerLayout

mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED); //关滑动的DrawerLayout

这种是click事件绑定的模式，代码自动生成关联todo

光在xml页面 加以下代码就已经可以实现侧滑菜单了

<?xml version="1.0" encoding="utf-8"?>  
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/DraLay" //这里的ID一定要写，后边用得到  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".Main3Activity">  
  
 <FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"> //FrameLayout里边就是页面的内容

<android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#ccc">  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:onClick="OpenDrewerLayout"  
 android:text="点击打开侧栏" />  
 </android.support.v7.widget.Toolbar>  
 </FrameLayout>  
  
 <LinearLayout //这里的控件可以是任何，但是要有 android:layout\_gravity="start" id后边也用得到  
 android:id="@+id/rights"  
 android:layout\_width="240dp"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:background="#f0f0f0">  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#ccc"  
 android:text="标题1" />  
 </LinearLayout>  
  
</android.support.v4.widget.DrawerLayout>

## 上拉刷新SwipeRefreshLayout

XML页面中必须是一个SwipeRefreshLayout嵌套ListView 或者Recycler

<android.support.v4.widget.SwipeRefreshLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:id="@+id/swipeLayout" >  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/ListView\_Id">  
 </ListView>  
</android.support.v4.widget.SwipeRefreshLayout>

Activity

swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeLayout); //下拉刷新控件的名称  
// swipeRefreshLayout.setColorSchemeResources(R.color.colorAccent); //刷新按钮的颜色  
// swipeRefreshLayout.setSize(SwipeRefreshLayout.DEFAULT); //设置进度View样式的大小 只有两个值DEFAULT （小）和LARGE（大）  
// swipeRefreshLayout.setProgressBackgroundColor(R.color.colorPrimaryDark); //刷新按钮的背景色  
// swipeRefreshLayout.setProgressViewEndTarget(true, 200); ////设置进度View下拉的结束点，scale 是指设置是否需要放大或者缩小动画  
// swipeRefreshLayout.setDistanceToTriggerSync(200); //设置触发刷新的距离  
  
 swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
 @Override  
 public void onRefresh() {  
 Log.i("ds","我进了下拉刷新");  
// swipeRefreshLayout.setRefreshing(false);  
// new Thread(new Runnable() {  
// @Override  
// public void run() {  
  
// try {  
// Thread.sleep(1500);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// 这里添加代码  
// swipeRefreshLayout.setRefreshing(false);  
// }  
// }).start();  
 }  
 });

最精简写法

swipeRefreshLayout = (SwipeRefreshLayout)findViewById(R.id.swipeLayout); //下拉刷新控件的名称  
swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
 @Override  
 public void onRefresh() {  
 Log.i("ds","我进了下拉刷新");  
 }  
});

## 高德地图

### 1）获取key

测试版好像都不影响有KEY就行



【获取 SHA1】

方法1：

家里的电脑 B1:EB:4C:26:2F:DB:62:D4:69:71:11:18:04:35:29:54:38:EF:A9:F9

如果提示keytool不存在

打开jdk /bin 找到 keytool.exe 家里是D:\all\Java\jdk1.8.0\_201\bin

打开它，在里边输入.\keytool.exe -list -keystore 路径\debug.keystore 家里是 .\keytool.exe -list -keystore C:\Users\diy\.android\debug.keystore

去.android文件夹子下 shift+右键打开Powershell中 输入keytool -v -list -keystore debug.keystore

提示输入密钥库密码，输入密码是看不见位数变多的，输就是了，开发模式默认密码是 android，发布模式的密码是为 apk 的 keystore 设置的密码。输入密钥后回车（如果没设置密码，可直接回车），此时可在控制台显示的信息中获取 Sha1 值

已经安装好android 的所有环境的话

1. WIN+R cmd打开命令行窗口
2. 在命令行窗口输入 cd .android 定位到 .android 文件夹
3. debug.keystore：命令为：keytool -list -v -keystore debug.keystore
4. 输了密码就可以看了

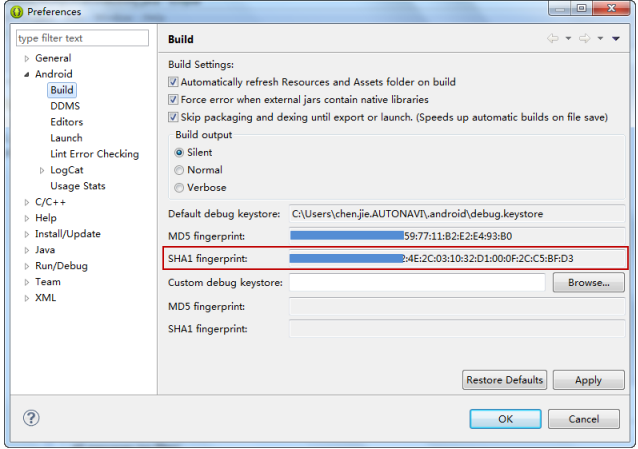
方法2：（此方法的eclipse需要安装ADT）

使用 adt 22 以上版本，可以在 eclipse 中直接查看。

Windows：依次在 eclipse 中打开 Window -> Preferances -> Android -> Build。

Mac：依次在 eclipse 中打开 Eclipse/ADT->Preferances -> Android -> Build。

在弹出的 Build 对话框中 “SHA1 fingerprint” 中的值即为 Android 签名证书的 SHA1 值，如下图所示：

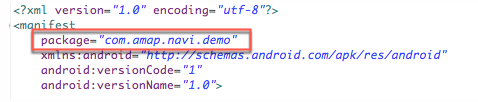


【获取 PackageName】

PackageName 即为建项目时的PackageName

也可以在项目中直接查看

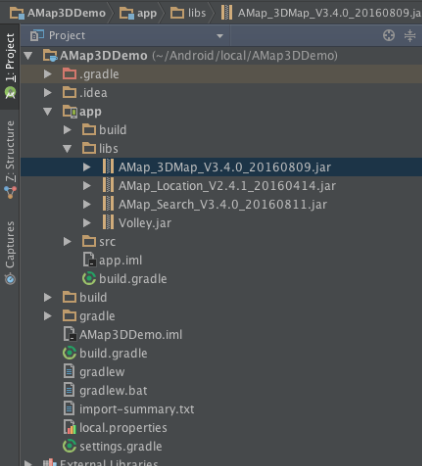
Android视图模式下 manifests-AndroidManifest.xml



### 文件的引入

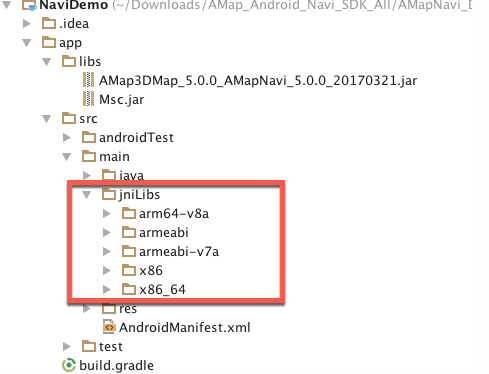
【添加jar文件】

将下载的地图 SDK 的 jar包复制到工程libs 目录下



【添加so库】

方法一：在main目录下创建文件夹jniLibs，将下载文件除了jar包，其它全部放在里边

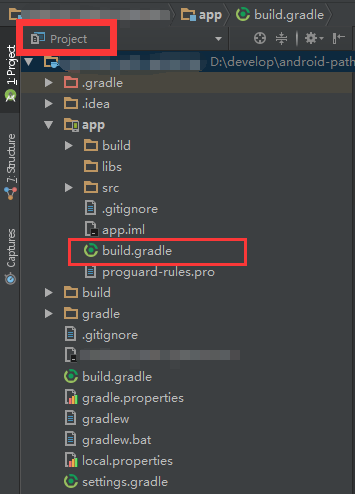


方法二：直接复制到 libs 目录，然后打开build.gradle，找到 asourceSets 标签，在里面增加一项配置，如图所示：



### 3）引入后的对应配置

在build.gradle中配置，build.gradle文件在Project目录中位置



【jar包的引入后配置】

implementation fileTree(dir: 'libs', include: ['\*.jar']) //引入libs文件夹下的所有jar包，一般默认有这句话

以下为引用解释

compile files('libs/Android\_Map3D\_SDK\_V6.8.0\_20190401.jar') //引入jar包…

//网上下的会报错

compile 'com.amap.api:3dmap:latest.integration' //网上下载最新版本

compile 'com.amap.api:3dmap:5.0.0' //网上下载指定版本

如果从网上直接下载

|  |  |
| --- | --- |
| **SDK** | **引入代码** |
| 3D地图 | compile 'com.amap.api:3dmap:latest.integration' |
| 2D地图 | compile 'com.amap.api:map2d:latest.integration' |
| 导航 | compile 'com.amap.api:navi-3dmap:latest.integration' |
| 搜索 | compile 'com.amap.api:search:latest.integration' |
| 定位 | compile 'com.amap.api:location:latest.integration' |

【armeabi引入后的配置】

这个跟5个包有关联，关手机只需要armeabi

android {

defaultConfig

{  
 ndk {  
 //设置支持的SO库架构（开发者可以根据需要，选择一个或多个平台的so）  
 abiFilters "armeabi", "armeabi-v7a", "arm64-v8a", "x86","x86\_64"  
 }

}  
}

### 4）AndroidManifest.xml文件key及各权限的配置

【配置权限】

在manifest标签下直接插入

//地图SDK（包含其搜索功能）需要的基础权限

<!--允许程序打开网络套接字-->  
<uses-permission android:name="android.permission.INTERNET" />  
<!--允许程序设置内置sd卡的写权限-->  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<!--允许程序获取网络状态-->  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<!--允许程序访问WiFi网络信息-->  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<!--允许程序读写手机状态和身份-->  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<!--允许程序访问CellID或WiFi热点来获取粗略的位置-->  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

【key的引入】

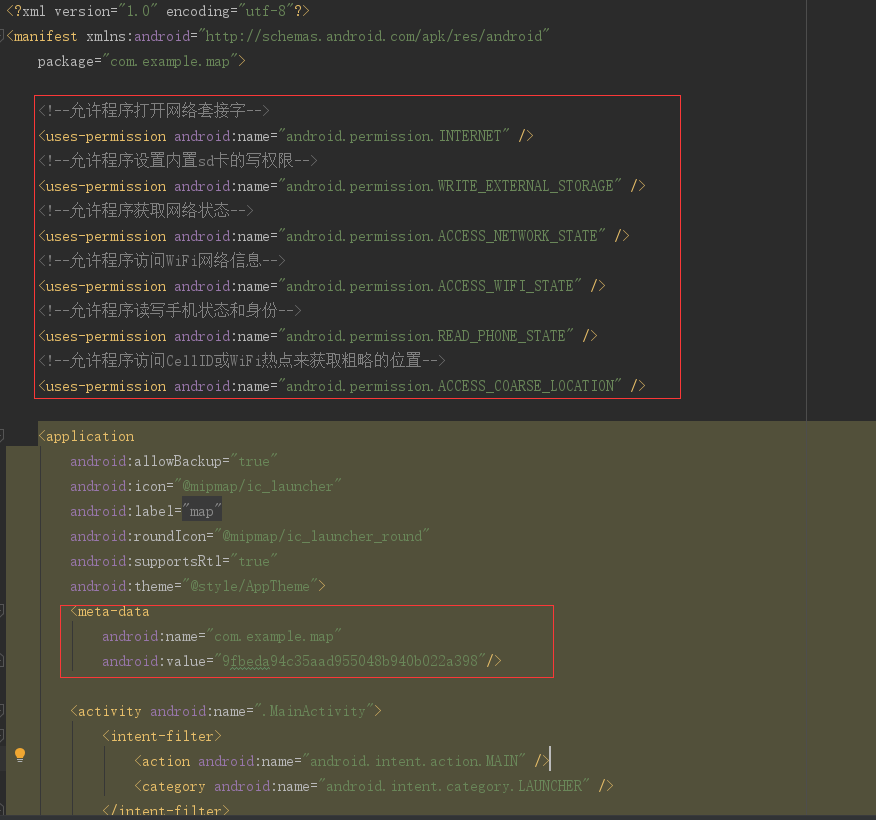


在application标签中加入如下内容：

<meta-data android:name="com.amap.api.v2.apikey" android:value="开发者申请的key">

</meta-data>

<meta-data  
 android:name="com.amap.api.v2.apikey"  
 android:value="9fbeda94c35aad955048b940b022a398"/>  
<service android:name="com.amap.api.location.APSService"></service>



### 5）初始化地图容器（layout）

<com.amap.api.maps.MapView

android:id="@+id/map"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

</com.amap.api.maps.MapView>

### 6）初始化地图Activity

MapView mMapView = (MapView) findViewById(R.id.map); //获取地图控件引用

mMapView.onCreate(savedInstanceState);// 此方法须覆写，虚拟机需要在很多情况下保存地图绘制的当前状态。

AMap aMap = mMapView.getMap(); //初始化地图控制器对象

aMap.setTrafficEnabled(true);// 显示实时交通状况>



### 7）Amap地图控制器

AMap aMap = mMapView.getMap(); //初始化地图控制器对象

aMap.setTrafficEnabled(true);// 显示实时交通状况>

aMap.showIndoorMap(true); //true：显示室内地图；false：不显示；

aMap.setMapLanguage(AMap.ENGLISH); // AMap.CHINESE 表示中文，即"zh\_cn", AMap.ENGLISH 表示英文，即"en"

### 7）比例尺（不生效）

setScaleControlsEnabled(boolean b);//控制比例尺控件是否显示

### 6）绘点

LatLng latLng = new LatLng(33.906901,116.397972);

final Marker marker = aMap.addMarker(new MarkerOptions()

.position(latLng)

.title("北京2")

.snippet("DefaultMarker")

);

marker.showInfoWindow();//关闭点击打开标题弹框事件，让点标题永久显示

marker.hideInfoWindow（）//让点的标题默认隐藏，点击后显示

类 MarkerOptions查看以下地址

https://a.amap.com/lbs/static/unzip/Android\_Map\_Doc/3D/com/amap/api/maps/model/MarkerOptions.html

### 7）绘线

List<LatLng> latLngs = new ArrayList<LatLng>();

latLngs.add(new LatLng(39.999391,116.135972));

latLngs.add(new LatLng(39.898323,116.057694));

latLngs.add(new LatLng(39.900430,116.265061));

latLngs.add(new LatLng(39.955192,116.140092));

Polyline polyline =aMap.addPolyline(new PolylineOptions().

setUseTexture(false) //是否使用纹理贴图

.addAll(latLngs)

.width(20)

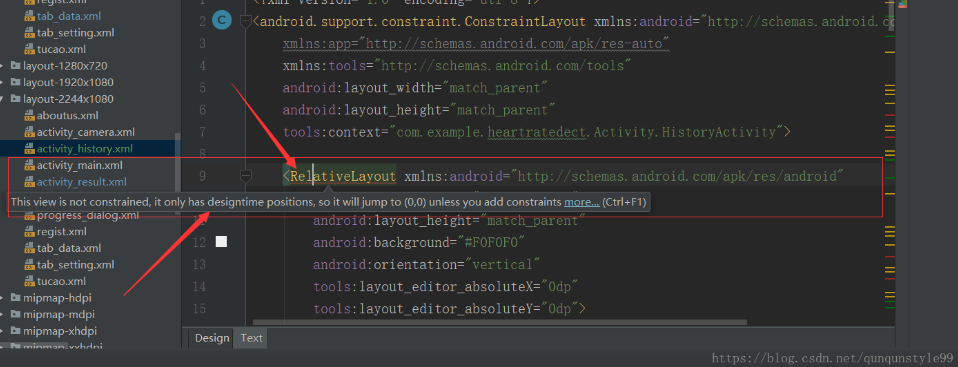
.color(Color.argb(255, 255, 1, 1)));

### 8）沿着轨迹移动

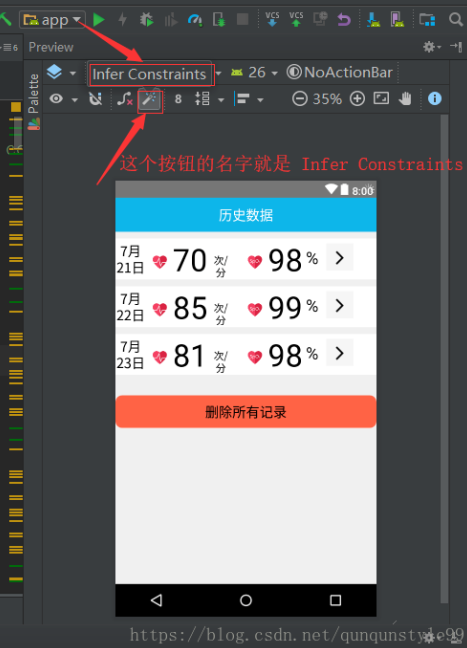
List<LatLng> latLngs = new ArrayList<LatLng>();  
 latLngs.add(new LatLng(39.999391,116.135972));  
 latLngs.add(new LatLng(39.898323,116.057694));  
 latLngs.add(new LatLng(39.900430,116.265061));  
 latLngs.add(new LatLng(39.955192,116.140092));  
 Polyline polyline =aMap.addPolyline(new PolylineOptions().  
 setUseTexture(false) //是否使用纹理贴图  
 .addAll(latLngs)  
 .width(20)  
 .color(Color.argb(255, 255, 1, 1)));  
   
  
 List<LatLng> points = latLngs;  
 LatLngBounds bounds = new LatLngBounds(points.get(0), points.get(points.size() - 2));  
 aMap.animateCamera(CameraUpdateFactory.newLatLngBounds(bounds, 50));  
  
 SmoothMoveMarker smoothMarker = new SmoothMoveMarker(aMap);  
// 设置滑动的图标  
 smoothMarker.setDescriptor(BitmapDescriptorFactory.fromResource(R.drawable.car));  
  
 LatLng drivePoint = points.get(0);  
 Pair<Integer, LatLng> pair = SpatialRelationUtil.calShortestDistancePoint(points, drivePoint);  
 points.set(pair.first, drivePoint);  
 List<LatLng> subList = points.subList(pair.first, points.size());  
  
// 设置滑动的轨迹左边点  
 smoothMarker.setPoints(subList);  
// 设置滑动的总时间  
 smoothMarker.setTotalDuration(40);  
// 开始滑动  
 smoothMarker.startSmoothMove();

### 9）在地图上设置功能按钮

浮动按钮在ConstraintLayout布局中，添加了button等容易报下边这个错：



解决办法：选中报错那个控件，在设计预览界面，找到“小魔法棒”的图标，点一下就解决了



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

<androidx.constraintlayout.widget.ConstraintLayout 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">

<com.amap.api.maps.MapView

android:id="@+id/map"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:visibility="visible"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"></com.amap.api.maps.MapView>

<Button

android:id="@+id/button"

android:layout\_width="40dp"

android:layout\_height="40dp"

android:layout\_marginTop="464dp"

android:layout\_marginEnd="128dp"

android:onClick="aa"

android:text="定位"

android:visibility="visible"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

### 10）离线地图

设置权限（这里一般设地图权限的时候就加进去了，不需要再弄一次）

权限包涵网络和SD卡读写权限

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

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

注册Activity

<activity android:name="com.amap.api.maps.offlinemap.OfflineMapActivity"

android:screenOrientation="portrait" />

启动离线地图组件

startActivity(new Intent(this.getApplicationContext(),

com.amap.api.maps.offlinemap.OfflineMapActivity.class));

自定义地图

OfflineMapManager amapManager = new OfflineMapManager(this, this);

实例化此变量的时候，需引入三个实现方法

@Override

public void onDownload(int status, int completeCode, String downName) {

System.out.println("status");

System.out.println("completeCode");

System.out.println("downName");

switch (status) {

case OfflineMapStatus.SUCCESS:

//下载成功

break;

case OfflineMapStatus.LOADING:

//加载中

break;

case OfflineMapStatus.UNZIP:

//解压

break;

case OfflineMapStatus.WAITING:

//等待下载

break;

case OfflineMapStatus.PAUSE:

//暂停

break;

case OfflineMapStatus.STOP:

//停止

break;

case OfflineMapStatus.ERROR:

//错误

break;

case OfflineMapStatus.EXCEPTION\_AMAP:

//地图异常

break;

case OfflineMapStatus.EXCEPTION\_NETWORK\_LOADING:

//网络加载错误

break;

case OfflineMapStatus.EXCEPTION\_SDCARD:

//SD存储错误

break;

default:

break;

}

}

@Override

public void onCheckUpdate(boolean b, String s) {

}

@Override

public void onRemove(boolean b, String s, String s1) {

}

### 11）放大缩小的位置

只提供了右中和右下两种

aMap.getUiSettings().setZoomPosition(AMapOptions.ZOOM\_POSITION\_RIGHT\_CENTER);

aMap.getUiSettings().setZoomPosition(AMapOptions.ZOOM\_POSITION\_RIGHT\_BUTTOM);

### 12）去logo的方法

只提供了右中和右下两种

aMap.getUiSettings().setLogoBottomMargin(-100);

### 12）清除地图元素

aMap.clear();

### 13）定位到地图某个位置的两个方法

方法一

AllAmap.myMap.moveCamera(CameraUpdateFactory.zoomTo(10)); //设置显示比例

LatLng latLng = new LatLng(39.898323,116.057694);

AllAmap.myMap. moveCamera(CameraUpdateFactory.changeLatLng(latLng));

方法二

AllAmap.myMap.moveCamera(CameraUpdateFactory.newCameraPosition(new CameraPosition(

new LatLng(39.898323,116.057694), 18, 30, 30)));

## 高德地图的定位

引入带高德带定位的jar包，全部功能集成在一起的的也好，只有导航功能的也好，无所谓，后边也可以随时替换，有时候一下找不到，重启下AS

### AndroidManifest.xml的配置

切记开权限啊

在application标签中声明meta-data组件：

配置key，这里android:name必须这样写，key是自己申请到的

<meta-data android:name="com.amap.api.v2.apikey" android:value="您的Key"></meta-data>

在application标签中声明service组件：

<service android:name="com.amap.api.location.APSService"></service>

配置权限，这里的权限最好还上调地图那些基本的权限:

<!--用于进行网络定位-->

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"></uses-permission>

<!--用于访问GPS定位-->

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"></uses-permission>

<!--获取运营商信息，用于支持提供运营商信息相关的接口-->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"></uses-permission>

<!--用于访问wifi网络信息，wifi信息会用于进行网络定位-->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"></uses-permission>

<!--这个权限用于获取wifi的获取权限，wifi信息会用来进行网络定位-->

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"></uses-permission>

<!--用于访问网络，网络定位需要上网-->

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

<!--用于读取手机当前的状态-->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"></uses-permission>

<!--写入扩展存储，向扩展卡写入数据，用于写入缓存定位数据-->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"></uses-permission>

<!--用于申请调用A-GPS模块-->

<uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS"></uses-permission>

<!--用于申请获取蓝牙信息进行室内定位-->

<uses-permission android:name="android.permission.BLUETOOTH"></uses-permission>

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"></uses-permission>

完整示例

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.map">

<!-- 允许程序打开网络套接字 -->

<uses-permission android:name="android.permission.INTERNET" /> <!-- 允许程序设置内置sd卡的写权限 -->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" /> <!-- 允许程序获取网络状态 -->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" /> <!-- 允许程序访问WiFi网络信息 -->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" /> <!-- 允许程序读写手机状态和身份 -->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" /> <!-- 允许程序访问CellID或WiFi热点来获取粗略的位置 -->

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" /> <!-- 用于进行网络定位 -->

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" /> <!-- 用于访问GPS定位 -->

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" /> <!-- 用于获取运营商信息，用于支持提供运营商信息相关的接口 -->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" /> <!-- 用于访问wifi网络信息，wifi信息会用于进行网络定位 -->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" /> <!-- 用于获取wifi的获取权限，wifi信息会用来进行网络定位 -->

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" /> <!-- 用于访问网络，网络定位需要上网 -->

<uses-permission android:name="android.permission.INTERNET" /> <!-- 用于读取手机当前的状态 -->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" /> <!-- 用于写入缓存数据到扩展存储卡 -->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" /> <!-- 用于申请调用A-GPS模块 -->

<uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" /> <!-- 用于申请获取蓝牙信息进行室内定位 -->

<uses-permission android:name="android.permission.BLUETOOTH" />

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

<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">

<meta-data

android:name="com.amap.api.v2.apikey"

android:value="2fad362a39937e8c17cc6408fed2edc6"></meta-data>

<service android:name="com.amap.api.location.APSService" />

<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>

### 2）activity\_main.xml的配置

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

<androidx.constraintlayout.widget.ConstraintLayout 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">

<com.amap.api.maps.MapView

android:id="@+id/map"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:visibility="visible"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"></com.amap.api.maps.MapView>

<Button

android:id="@+id/button"

android:layout\_width="40dp"

android:layout\_height="40dp"

android:layout\_marginTop="464dp"

android:layout\_marginEnd="128dp"

android:onClick="aa"

android:text="定位"

android:visibility="visible"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

### 3）MainActivity的配置

package com.example.map;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Color;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Toast;

import com.amap.api.location.AMapLocation;

import com.amap.api.maps.AMap;

import com.amap.api.maps.CameraUpdateFactory;

import com.amap.api.maps.LocationSource;

import com.amap.api.maps.MapView;

import com.amap.api.maps.model.BitmapDescriptorFactory;

import com.amap.api.maps.model.LatLng;

import com.amap.api.maps.model.MyLocationStyle;

import com.example.map.util.LocationUtil;

public class MainActivity extends AppCompatActivity implements LocationSource {

private MapView mapView;

private AMap aMap;

private LocationSource.OnLocationChangedListener mListener = null;//定位监听器

private LocationUtil locationUtil;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mapView = (MapView) findViewById(R.id.map);

mapView.onCreate(savedInstanceState);

init();

}

private void init() {

if(aMap == null){

aMap = mapView.getMap();

}

setLocationCallBack();

//设置定位监听

aMap.setLocationSource(this);

//设置缩放级别

aMap.moveCamera(CameraUpdateFactory.zoomTo(15));

//显示callBack方法里的事件

aMap.setMyLocationEnabled(true);

//显示定位的图标

aMap.getUiSettings().setMyLocationButtonEnabled(true);

//启动的时候就定位

// activate(mListener);

}

public void aa(View view){

aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(aMap.getMyLocation().getLatitude(), aMap.getMyLocation().getLongitude())));

Toast.makeText(MainActivity.this,String.valueOf(aMap.getMyLocation().getLatitude()),Toast.LENGTH\_LONG).show();

}

private void setLocationCallBack(){

locationUtil = new LocationUtil();

locationUtil.setLocationCallBack(new LocationUtil.ILocationCallBack() {

@Override

public void callBack(String str,double lat,double lgt,AMapLocation aMapLocation) {

Log.e("回调信息",String.valueOf(lat)+String.valueOf(lgt)+str);

//以下是定义定位图标

MyLocationStyle myLocationStyle = new MyLocationStyle();

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE\_NO\_CENTER);

myLocationStyle.myLocationIcon(BitmapDescriptorFactory.

fromResource(R.mipmap.ic\_launcher\_round));// 自定义定位蓝点图标

myLocationStyle.strokeColor(Color.argb(0, 0, 0, 0));// 自定义精度范围的圆形边框颜色

myLocationStyle.radiusFillColor(Color.argb(0, 0, 0, 0));//圆圈的颜色,设为透明的时候就可以去掉园区区域了

aMap.setMyLocationStyle(myLocationStyle);

//核心方法

mListener.onLocationChanged(aMapLocation);

//自己可以设置的飞点及绘图标，但MyLocationStyle明显更好，建议废弃

//aMap.moveCamera(CameraUpdateFactory.changeLatLng(new LatLng(lat,lgt))); //根据获取的经纬度，将地图移动到定位位置

//aMap.addMarker(locationUtil.getMarkerOption(str,lat,lgt));//添加定位图标 这个是locationUtil类中自己定义的设置图标的方法

}

});

}

@Override

public void activate(OnLocationChangedListener onLocationChangedListener) {

mListener = onLocationChangedListener;

locationUtil.startLocate(getApplicationContext());

}

@Override

public void deactivate() {

mListener = null;

}

@Override

public void onPointerCaptureChanged(boolean hasCapture) {

}

@Override

protected void onPause() {

super.onPause();

//暂停地图的绘制

mapView.onPause();

}

@Override

protected void onDestroy() {

super.onDestroy();

//销毁地图

mapView.onDestroy();

}

@Override

protected void onResume() {

super.onResume();

//重新绘制加载地图

mapView.onResume();

}

@Override

public void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

mapView.onSaveInstanceState(outState);

}

}

### 4）util/LocationUtil的配置

package com.example.map.util;

import android.content.Context;

import android.util.Log;

import com.amap.api.location.AMapLocation;

import com.amap.api.location.AMapLocationClient;

import com.amap.api.location.AMapLocationClientOption;

import com.amap.api.location.AMapLocationListener;

import com.amap.api.maps.model.BitmapDescriptorFactory;

import com.amap.api.maps.model.LatLng;

import com.amap.api.maps.model.MarkerOptions;

import com.example.map.R;

import java.text.SimpleDateFormat;

import java.util.Date;

public class LocationUtil implements AMapLocationListener {

private AMapLocationClient aMapLocationClient;

private AMapLocationClientOption clientOption;

private ILocationCallBack callBack;

public void startLocate(Context context){

aMapLocationClient = new AMapLocationClient(context);

//设置监听回调

aMapLocationClient.setLocationListener(this);

//初始化定位参数

clientOption = new AMapLocationClientOption(); //核心方法

clientOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.Battery\_Saving);//设置定位模式 Hight\_Accuracy：高精度定位模式；Device\_Sensors：仅设备定位模式；Battery\_Saving：低功耗定位模式；

clientOption.setNeedAddress(true);//返回经纬度同时返回地址描述

clientOption.setOnceLocation(true); //true表示单次定位 false 表示连续定

//设置是否强制刷新WIFI，默认为强制刷新

clientOption.setWifiActiveScan(true);

//设置是否允许模拟位置,默认为false，不允许模拟位置

clientOption.setMockEnable(false);

//设置定位间隔

// clientOption.setInterval(2000000);

// aMapLocationClient.setLocationOption(clientOption);

aMapLocationClient.startLocation();

// aMapLocationClient.stopLocation();

}

/\*\*

\* 完成定位回调

\*/

@Override

public void onLocationChanged(AMapLocation aMapLocation) {

if(aMapLocation != null){

if(aMapLocation.getErrorCode() == 0){

//定位成功完成回调

String country = aMapLocation.getCountry();

String province = aMapLocation.getProvince();

String city = aMapLocation.getCity();

String district = aMapLocation.getDistrict();

String street = aMapLocation.getStreet();

String streetNum=aMapLocation.getStreetNum();//街道门牌号信息

String floor=aMapLocation.getFloor();//获取当前室内定位的楼层

//获取定位时间

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = new Date(aMapLocation.getTime());

double lat = aMapLocation.getLatitude();

double lgt = aMapLocation.getLongitude();

//如果没有设置myLocationStyle的模式，滑动地图总是会定位到原点，本质上是由于不管设置单次定位还是连续定位，都会一直在定位，这里需要的时候可以判断一下，但最好不用

// if(aMapLocationClient.isStarted()){

// aMapLocationClient.stopLocation();

// }

callBack.callBack(country + province + city + district + street+streetNum+floor+ df.format(date),lat,lgt,aMapLocation);

}else{

//显示错误信息ErrCode是错误码，errInfo是错误信息，详见错误码表。

Log.e("地图调用出错", "location Error, ErrCode:"

+ aMapLocation.getErrorCode() + ", errInfo:"

+ aMapLocation.getErrorInfo());

}

}

}

/\*\*

\* 自定义图标

\* 此方法跟myLocationStyle相比，无法选择方向等，建议废弃

\* @return

\*/

public MarkerOptions getMarkerOption(String str, double lat, double lgt){

MarkerOptions markerOptions = new MarkerOptions();

markerOptions.icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic\_launcher));

markerOptions.position(new LatLng(lat,lgt));

markerOptions.title(str);

markerOptions.snippet("纬度:" + lat + " 经度:" + lgt);

markerOptions.period(100);

return markerOptions;

}

public interface ILocationCallBack{

void callBack(String str,double lat,double lgt,AMapLocation aMapLocation);

}

public void setLocationCallBack(ILocationCallBack callBack){

this.callBack = callBack;

}

}

### 5）新版辅助H5页面定位

WebView的引用可以参照前文

如果是远程地址别忘了配置权限<uses-permission android:name="android.permission.INTERNET"/>

Activity中WebView的实现

WebView webView=(WebView) findViewById(R.id.webView); //webView的ID

locationUtil.aMapLocationClient.startAssistantLocation(webView);

webView.loadUrl("https://www.ynmap.cn/1.html"); //网页地址

WebSettings webSettings = webView.getSettings();

// 允许webview执行javaScript脚本

webSettings.setJavaScriptEnabled(true);

//设置WebView属性，能够执行Javascript脚本

webView.setWebViewClient(new WebViewClient() {

public void onPageFinished(WebView view, String url) {

super.onPageFinished(view, url);

}

public void onPageStarted(WebView view, String url, Bitmap favicon) {

super.onPageStarted(view, url, favicon);

}

});

webView.setWebChromeClient(new WebChromeClient() {

// 处理javascript中的alert

public boolean onJsAlert(WebView view, String url, String message,

final JsResult result) {

return true;

};

// 处理javascript中的confirm

public boolean onJsConfirm(WebView view, String url,

String message, final JsResult result) {

return true;

};

// 处理定位权限请求

@Override

public void onGeolocationPermissionsShowPrompt(String origin,

GeolocationPermissions.Callback callback) {

callback.invoke(origin, true, false);

super.onGeolocationPermissionsShowPrompt(origin, callback);

}

@Override

// 设置网页加载的进度条

public void onProgressChanged(WebView view, int newProgress) {

MainActivity.this.getWindow().setFeatureInt( //MainActivity为当前Activity

Window.FEATURE\_PROGRESS, newProgress \* 100);

super.onProgressChanged(view, newProgress);

}

// 设置应用程序的标题title

public void onReceivedTitle(WebView view, String title) {

super.onReceivedTitle(view, title);

}

});

Html页面

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>RectAreaLight 矩形平面光源</title>

<script type="text/javascript" src = 'https://webapi.amap.com/maps?v=1.4.4&test=true&key=2fad362a39937e8c17cc6408fed2edc6&plugin=AMap.ToolBar,AMap.IndoorMap'></script>

</head>

<body>

<div class="log">日志</div>

<script type="text/javascript">

var log = document.getElementsByClassName('log')[0];

var map = new AMap.Map('container',{

resizeEnable: true,

zoom: 10,

center: [116.43, 39.9]

});

AMap.plugin(['AMap.Geolocation'],

function(){

geo = new AMap.Geolocation({

useNative:true

})

geo.on('complete',function(e){

log.innerHTML+=e.info+' '+e.position+' '+e.message.split('.')[0]+'</br>'

})

geo.on('fail',function(e){

log.innerHTML+=e.info+e.message.split('.')[0]+'</br>'

})

// map.addControl(geo);

});

</script>

//获取位置按钮

<button class = 'bt' onclick='geo.getCurrentPosition()'>获取位置</button>

<button class = 'bt' onclick='geo.watchPosition()'>连续定位</button>

<button class = 'bt' onclick='geo.clearWatch()'>停止定位</button>

</body>

</html>

### 5）5.0版本以后定位写法

MyLocationStyle myLocationStyle;

myLocationStyle = new MyLocationStyle();//初始化定位蓝点样式类

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE\_NO\_CENTER);//连续定位、蓝点不会移动到地图中心点，定位点依照设备方向旋转，并且蓝点会跟随设备移动。

myLocationStyle.interval(2000); //设置连续定位模式下的定位间隔，只在连续定位模式下生效，单次定位模式下不会生效。单位为毫秒。

aMap.setMyLocationStyle(myLocationStyle);//设置定位蓝点的Style

aMap.getUiSettings().setMyLocationButtonEnabled(true); //设置默认定位按钮是否显示，非必需设置。

aMap.setMyLocationEnabled(true);// 设置为true表示启动显示定位蓝点，false表示隐藏定位蓝点并不进行定位，默认是false。

myLocationType可用的选择，一般选倒数第三种

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_SHOW);//只定位一次。

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATE) ;//定位一次，且将视角移动到地图中心点。

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_FOLLOW) ;//连续定位、且将视角移动到地图中心点，定位蓝点跟随设备移动。（1秒1次定位）

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_MAP\_ROTATE);//连续定位、且将视角移动到地图中心点，地图依照设备方向旋转，定位点会跟随设备移动。（1秒1次定位）

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE);//连续定位、且将视角移动到地图中心点，定位点依照设备方向旋转，并且会跟随设备移动。（1秒1次定位）默认执行此种模式。

//以下三种模式从5.1.0版本开始提供

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE\_NO\_CENTER);//连续定位、蓝点不会移动到地图中心点，定位点依照设备方向旋转，并且蓝点会跟随设备移动。

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_FOLLOW\_NO\_CENTER);//连续定位、蓝点不会移动到地图中心点，并且蓝点会跟随设备移动。

myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_MAP\_ROTATE\_NO\_CENTER);//连续定位、蓝点不会移动到地图中心点，地图依照设备方向旋转，并且蓝点会跟随设备移动。

控制是否显示定位蓝点

//方法自5.1.0版本后支持

MyLocationStyle showMyLocation(boolean visible)//设置是否显示定位小蓝点，用于满足只想使用定位，不想使用定位小蓝点的场景，设置false以后图面上不再有定位蓝点的概念，但是会持续回调位置信息。

自定义定位蓝点图标：

MyLocationStyle myLocationIcon(BitmapDescriptor myLocationIcon);//设置定位蓝点的icon图标方法，需要用到BitmapDescriptor类对象作为参数。

自定义定位蓝点图标的锚点：

锚点是指定位蓝点图标像素与定位蓝点坐标的关联点，例如需要将图标的左下方像素点与定位蓝点的经纬度关联在一起，通过如下方法传入（0.0,1.0）。图标左上点为像素原点。

MyLocationStyle anchor(float u, float v);//设置定位蓝点图标的锚点方法。

精度圆圈的自定义：

MyLocationStyle strokeColor(int color);//设置定位蓝点精度圆圈的边框颜色的方法。

MyLocationStyle radiusFillColor(int color);//设置定位蓝点精度圆圈的填充颜色的方法。

精度圈边框宽度自定义方法如下

MyLocationStyle strokeWidth(float width);//设置定位蓝点精度圈的边框宽度的方法。

定位的频次自定义：

定位频次修改只会在定位蓝点的连续定位模式下生效，定位蓝点支持连续定位的模式是：

MyLocationStyle.LOCATION\_TYPE\_FOLLOW ;//连续定位、且将视角移动到地图中心点，定位蓝点跟随设备移动。（默认1秒1次定位）

MyLocationStyle.LOCATION\_TYPE\_MAP\_ROTATE;//连续定位、且将视角移动到地图中心点，地图依照设备方向旋转，定位点会跟随设备移动。（默认1秒1次定位）

MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE;//连续定位、且将视角移动到地图中心点，定位点依照设备方向旋转，并且会跟随设备移动。（默认1秒1次定位）默认执行此种模式。

调整定位频次的方法如下：

MyLocationStyle interval(long interval);//设置定位频次方法，单位：毫秒，默认值：1000毫秒，如果传小于1000的任何值将按照1000计算。该方法只会作用在会执行连续定位的工作模式上。

## 高德地图的导航

### 项目创建

项目创建必须SDK 25以上（前期与上边相同）

### 文件的引入AndroidManifest.xml文件key及各权限的配置

AndroidManifest.xml中配置权限

<!--用于访问网络，网络定位需要上网-->

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

<!--写入扩展存储，向扩展卡写入数据，用于写入缓存定位数据-->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<!--用于进行网络定位--> <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<!--获取运营商信息，用于支持提供运营商信息相关的接口-->

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<!--用于访问GPS定位-->

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<!--用于读取手机当前的状态-->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<!--用于访问wifi网络信息，wifi信息会用于进行网络定位-->

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<!--这个权限用于获取wifi的获取权限，wifi信息会用来进行网络定位-->

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />

<!--这个权限用于允许程序在手机屏幕关闭后后台进程仍然运行-->

<uses-permission android:name="android.permission.WAKE\_LOCK"/>

<meta-data  
 android:name="com.amap.api.v2.apikey"  
 android:value="您的KEY" />

<service android:name="com.amap.api.location.APSService" />

引入导航插件

<activity android:name="com.amap.api.navi.AmapRouteActivity"  
 android:theme="@android:style/Theme.NoTitleBar"  
 android:configChanges="orientation|keyboardHidden|screenSize" />

### 文件的引入AndroidManifest.xml文件key及各权限的配置

类继承 implements INaviInfoCallback 该引入方法引入方法

<meta-data  
 android:name="com.amap.api.v2.apikey"  
 android:value="您的KEY" />

无起点终点

AmapNaviPage.getInstance().showRouteActivity(getApplicationContext(), new AmapNaviParams(null), MainActivity.this);

有起点有终点

LatLng p2 = new LatLng(39.917337, 116.397056);//故宫博物院  
LatLng p3 = new LatLng(39.904556, 116.427231);//北京站  
  
AmapNaviParams params = new AmapNaviParams(new Poi("北京站", p3, ""), null, new Poi("故宫博物院", p2, ""), AmapNaviType.DRIVER);  
params.setUseInnerVoice(true);  
AmapNaviPage.getInstance().showRouteActivity(getApplicationContext(), params, MainActivity.this);

无起点有终点-起点位置会自动变成当前位置

LatLng p2 = new LatLng(25.0372383400, 102.7125281100);//昆明南屏街家乐福

AmapNaviPage.getInstance().showRouteActivity(getApplicationContext(), new AmapNaviParams(null, null, new Poi("南屏街家乐福", p2, ""), AmapNaviType.DRIVER), MainActivity.this);

## 扫码（可做演示用）

build.gradle 中应用包

implementation 'com.google.zxing:core:3.3.2'

AndroidManifest.xml 中配置页面

<activity android:name="zxing.activity.CaptureActivity">  
 <intent-filter>  
 <action android:name="zxing" />  
 <category android:name="android.intent.category.DEFAULT" />  
 </intent-filter>  
</activity>

Intent intent = new Intent(getActivity().getBaseContext(), CaptureActivity.class);  
startActivityForResult(intent, Constants.REQ\_QR\_CODE);

activity用getBaseContext(), Fragment用 getActivity().getBaseContext()

## 指纹识别

指纹识别的使用

导入模块 biometriclib

如果sdk版本不同要改了一致

private Button mButton;

private BiometricPromptManager mManager;

private TextView mTextView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mManager = BiometricPromptManager.from(this);

mTextView = findViewById(R.id.text\_view);

StringBuilder stringBuilder = new StringBuilder();

stringBuilder.append("SDK version is "+ Build.VERSION.SDK\_INT);

stringBuilder.append("\n");

stringBuilder.append("isHardwareDetected : "+mManager.isHardwareDetected());

stringBuilder.append("\n");

stringBuilder.append("hasEnrolledFingerprints : "+mManager.hasEnrolledFingerprints());

stringBuilder.append("\n");

stringBuilder.append("isKeyguardSecure : "+mManager.isKeyguardSecure());

stringBuilder.append("\n");

mTextView.setText(stringBuilder.toString());

mButton = findViewById(R.id.button);

mButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (mManager.isBiometricPromptEnable()) {

mManager.authenticate(new BiometricPromptManager.OnBiometricIdentifyCallback() {

@Override

public void onUsePassword() {

Toast.makeText(MainActivity.this, "onUsePassword", Toast.LENGTH\_SHORT).show();

}

@Override

public void onSucceeded() {

Toast.makeText(MainActivity.this, "onSucceeded", Toast.LENGTH\_SHORT).show();

}

@Override

public void onFailed() {

Toast.makeText(MainActivity.this, "onFailed", Toast.LENGTH\_SHORT).show();

}

@Override

public void onError(int code, String reason) {

Toast.makeText(MainActivity.this, "onError", Toast.LENGTH\_SHORT).show();

}

@Override

public void onCancel() {

Toast.makeText(MainActivity.this, "onCancel", Toast.LENGTH\_SHORT).show();

}

});

} else {

Toast.makeText(MainActivity.this, "此手机不支持此功能", Toast.LENGTH\_SHORT).show();

}

}

});

}

## Notification消息推送 //todo

# 插件的使用

## 各种形状ImageView

git地址

<https://github.com/siyamed/android-shape-imageview>

引入

implementation 'com.github.siyamed:android-shape-imageview:0.9.+@aar'

使用：圆形，其它到git地址中看

<com.github.siyamed.shapeimageview.CircularImageView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:src="@drawable/neo"

app:siBorderWidth="6dp"

app:siBorderColor="@color/darkgray"/>

## 底部带弧形的布局

git地址

<https://github.com/florent37/ArcLayout>

引入

implementation 'com.github.florent37:arclayout:1.0.3'

使用：

<com.github.florent37.arclayout.ArcLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 app:arc\_cropDirection="cropOutside" // cropInside内凹

app:arc\_height="30dp"  
 app:arc\_position="bottom"  
 android:elevation="5dp"  
 >  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:scaleType="centerCrop"  
 android:src="@drawable/me" />  
  
</com.github.florent37.arclayout.ArcLayout>

## 视觉差

git地址

<https://github.com/flavioarfaria/KenBurnsView>

引入

implementation 'com.flaviofaria:kenburnsview:1.0.7'

使用：（这个和上一个配合使用效果很好）

<com.flaviofaria.kenburnsview.KenBurnsView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:scaleType="centerCrop"  
 android:src="@drawable/me"  
 />

## 带有角标的TextView

git地址

<https://github.com/HeZaiJin/SlantedTextView>

引入

implementation 'com.haozhang.libary:android-slanted-textview:1.2'

使用：（这个和上一个配合使用效果很好）

<com.haozhang.lib.SlantedTextView  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:gravity="center"  
 app:slantedBackgroundColor="#ff4f02"  
 app:slantedLength="20dp"  
 app:slantedMode="left\_triangle"  
 app:slantedText="已完成"  
 app:slantedTextColor="#fff"  
 app:slantedTextSize="10sp"  
 tools:ignore="MissingConstraints" />

java动态设置

SlantedTextView stv = (SlantedTextView) findViewById(R.id.test);

stv.setText("PHP")

.setTextColor(Color.WHITE)

.setSlantedBackgroundColor(Color.BLACK)

.setTextSize(18)

.setSlantedLength(50)

.setMode(SlantedTextView.MODE\_LEFT);

## 文字带淡入淡出动画的TextView

git地址

<https://github.com/matthewrkula/SecretTextView>

引入

implementation 'com.haozhang.libary:android-slanted-textview:1.2'

使用：（这个和上一个配合使用效果很好）

<com.haozhang.lib.SlantedTextView  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:gravity="center"  
 app:slantedBackgroundColor="#ff4f02"  
 app:slantedLength="20dp"  
 app:slantedMode="left\_triangle"  
 app:slantedText="已完成"  
 app:slantedTextColor="#fff"  
 app:slantedTextSize="10sp"  
 tools:ignore="MissingConstraints" />

## 下拉刷新[SmartRefreshLayout](https://github.com/scwang90/SmartRefreshLayout)

git地址

<https://github.com/scwang90/SmartRefreshLayout>

引入

implementation 'com.scwang.smartrefresh:SmartRefreshLayout:1.1.2' //1.0.5及以前版本的老用户升级需谨慎，API改动过大

implementation 'com.scwang.smartrefresh:SmartRefreshHeader:1.1.2' //没有使用特殊Header，可以不加这行

如果使用 AndroidX 在 gradle.properties 中添加

android.useAndroidX=true

android.enableJetifier=true

.在XML布局文件中添加 SmartRefreshLayout

有 header 和 footer 是经典模式

<com.scwang.smartrefresh.layout.SmartRefreshLayout

xmlns:app="http://schemas.android.com/apk/res-auto"

android:id="@+id/refreshLayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#444444"

app:srlPrimaryColor="#444444"

app:srlAccentColor="@android:color/white"

app:srlEnablePreviewInEditMode="true">

<!--srlAccentColor srlPrimaryColor 将会改变 Header 和 Footer 的主题颜色-->

<!--srlEnablePreviewInEditMode 可以开启和关闭预览功能-->

<com.scwang.smartrefresh.layout.header.ClassicsHeader

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="@dimen/dimenPaddingCommon"

android:background="@android:color/white"

android:text="@string/description\_define\_in\_xml"/>

<com.scwang.smartrefresh.layout.footer.ClassicsFooter

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"/>

</com.scwang.smartrefresh.layout.SmartRefreshLayout>

在 Activity 或者 Fragment 中添加代码

RefreshLayout refreshLayout = (RefreshLayout)findViewById(R.id.refreshLayout);

refreshLayout.setOnRefreshListener(new OnRefreshListener() {

@Override

public void onRefresh(RefreshLayout refreshlayout) {

refreshlayout.finishRefresh(2000/\*,false\*/);//传入false表示刷新失败

}

});

refreshLayout.setOnLoadMoreListener(new OnLoadMoreListener() {

@Override

public void onLoadMore(RefreshLayout refreshlayout) {

refreshlayout.finishLoadMore(2000/\*,false\*/);//传入false表示加载失败

}

});