宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

Android-Activity

1.Activity 简介（★★）

Activity 是 Android 四大组件之一，它用于展示界面。Activity 是一个应用程序组件，提供一个屏幕，用户可以用来交互为了完成某项任务。Activity 中所有操作都与用户密切相关，是一个负责与用户交互的组件，可以通过 setContentView(View)来显示指定控件。

在一个 android 应用中，一个 Activity 通常就是一个单独的屏幕，它上面可以显示一些控件也可以监听并处理用户的事件做出响应。Activity 之间通过 Intent 进行通信。

2.Activity 之间的跳转（★★★★）

Activity 之间的跳转分为 2 种：

- 显式跳转：在可以引用到那个类，并且可以引用到那个类的字节码时可以使用。一般用于自己程序的内部。显式跳转不可以跳转到其他程序的页面中。
- 隐式跳转：可以在当前程序跳转到另一个程序的页面。隐式跳转不需要引用到那个类,但是必须得知道那个界面的动作(action)和信息(category)。

Activity 之间通过 Intent 进行通信。Intent 即意图，用于描述一个页面的信息,同时也是一个数据的载体。

案例-显式跳转：显式跳转必须知道并且能够使用要跳转的 Activity 的字节码，所以显式跳转一般只能用于自己程序的内部的跳转，而不能跳转到其他程序的 Activity。

Tips：为了方便演示，我们创建一个新的 Android 工程《Activity 跳转》。然后创建

两个 Activity 类，分别为 FirstActivity，和 SecondActivity，并在 Android 工程清单文件

中声明这两个 Activity 类。工程清单中添加 Activity 配置如下。

```
<activity
           android:name="com.itheima.activitySkip.FirstActivity"
           android:label="@string/app_name" >
 <intent-filter>
<action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name="com.itheima.activitySkip.SecondActivity"/>
```

修改 FirstActivity 布局文件（activity_first.xml），并将该布局文件复制并修改名字为

activity_second.xml 作为 SecondActivity 的布局文件。

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

    <TextView
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#ff0000"
        android:text="我是第一个Activity" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="跳转到第二个Activity"
        android:onClick="skip2Second"
        />
</LinearLayout>
```

2.1 显式跳转

编写 FirstActivity 类：

```
package com.itheima.activitySkip;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class FirstActivity extends Activity {

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

    public void skip2Second(View view){
        //创建一个 Intent 对象，并传递当前对象（Context 对象）和要跳转的
Activity 类字节码
        Intent intent = new Intent(this, SecondActivity.class);
        //启动第二个 Activity
        startActivity(intent);
    }
}
```

运行上面的工程，点击按钮，发现成功跳转到了第二个 Activity 页面。

2.2 隐式跳转

隐式跳转可以跳转到其他程序的 Activity，只要知道 Activity 的动作(action)以及信息(category)。因此，能够被隐式跳转的 Activity，在清单文件中声明时必须指定动作和信息这两个属性。

1 修改工程清单文件中 SecondActivity 的配置信息。

```
 <activity android:name="com.itheima.activitySkip.SecondActivity">
          <!-- 配置意图过滤器 -->
          <intent-filter >
<!-- 在意图过滤器中设置 action 和 category，当有匹配的 action 和 category
的时候启动该 Activity。这里使用 Android 提供的默认 category 即可 -->
<action android:name="com.itheima.activitySkip.SecondActivity"/>
 <category android:name="android.intent.category.DEFAULT"/>
          </intent-filter>
          </activity>
```

2 修改在 2.1 章节中 FirstActivity 类

```
//创建一个 Intent 对象
      Intent intent = new Intent();
      //设置 Action
      intent.setAction("com.itheima.activitySkip.SecondActivity");
      //对于 android.intent.category.DEFAULT 类型的信息为 Android 系统
默认的信息，省略也可以
      intent.addCategory("android.intent.category.DEFAULT");
      //启动 Activity
      startActivity(intent);
```

执行上面的代码，发现实现了 Activity 的跳转。

Tips：若清单文件中的Activity声明为:

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

则此 Activity 将作为程序的入口，有几个作为入口的 Activity，apk 文件安装的时候就会生成几个图标。

3.案例-Activity 隐式跳转（★★★）

Android 系统中本身存在很多的应用程序，打开系统源码，查看 packages 文件夹下的 apps 文件夹，里面存放着 Android 系统的这些应用程序。

若想跳转到这些应用程序中，只需打开对应应用程序的清单文件，找到其动作和信息，采用隐式跳转即可实现。

3.1 打开浏览器界面

自己将 Android 应用的源码安装在 :D:\AndroidSource_GB\AndroidSource_GB 中，打开 D:\AndroidSource_GB\AndroidSource_GB\packages\apps\Browser 目录，然后打开 AndroidManifest.xml 清单文件，找到用于启动浏览器的 intent-filter。

Tips：intent-filter 有多个，我们选择一个适合我们即可。

```
 <intent-filter>
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="http" />
            <data android:scheme="https" />
            <data android:scheme="about" />
            <data android:scheme="javascript" />
</intent-filter>
```

注意：为了方便演示我们直接修改本文档 2.1 章节中创建的工程，在 FirstActivity 的布局文件中添加一个按钮，点击该按钮打开一个浏览器。这里给出 FirstActivity 类核心方法的代码清单：

```
//跳转到浏览器界面
    public void skip2Browser(View view){
        //创建一个 Intent 对象
        Intent intent = new Intent();
        //设置 Action
        intent.setAction("android.intent.action.VIEW");
        //设置 category
        intent.addCategory("android.intent.category.BROWSABLE");
        //设置参数
        intent.setData(Uri.parse("http://www.itheima.com"));
        //启动 Activity
        startActivity(intent);
    }
```

运行上面代码，发现成功跳转到了浏览器界面，并打开了指定的网页。

3.2 打开短信发送界面

打开 Android 应用源码，找到

D:\AndroidSource_GB\AndroidSource_GB\packages\apps\Mms 目录，打开

AndroidManifest.xml 清单文件，找到 intent-filter。

```
 <intent-filter>
<action android:name="android.intent.action.VIEW" />
<action android:name="android.intent.action.SENDTO" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
         <data android:scheme="sms" />
        <data android:scheme="smsto" />
</intent-filter>
```

在本文档 2.1 章节中给 FirstActivity 布局文件添加一个按钮，并添加按钮事件。在该方法中

实现核心业务逻辑。

```
//跳转到发送短信界面
    public void skip2Mms(View view){
        //创建一个 Intent 对象
        Intent intent = new Intent();
        //设置 Action
        intent.setAction("android.intent.action.VIEW");
        //设置 category
        intent.addCategory("android.intent.category.BROWSABLE");
        //设置参数
        intent.setData(Uri.parse("sms:10086"));
        //设置短信内容
        intent.putExtra("sms_body", "301");
        //启动 Activity
        startActivity(intent);
    }
```

运行上面的代码，发现成功打开了短信发送界面。这里就不给出运行效果图。

4.使用 Intent 传递数据（★★★★）

Intent 可传递的数据类型有：八大基本类型 ,数组,ArrayList<String>, Bundle 数

据捆，序列化接口(javabean)。

注意：Intent 传递的数据过多可能会造成跳转速度极慢甚至黑屏一会，不要用 Intent 传递过多的数据，会影响到应用程序的使用。

下面通过一个案例来演示 Intent 如何传递数据，我们依然使用 2.1 章节中的工程。需求：点击 FirstActivity 界面的按钮，将数据传递到 SecondActivity，并显示传递的数据。

1 修改 AndroidManifest.xml 文件中 SecondActivity 的 intent-filter 参数

```
<activity android:name="com.itheima.activitySkip.SecondActivity">
            <!-- 配置意图过滤器 -->
            <intent-filter >
                <!-- 在意图过滤器中设置 action 和 category，当有匹配的
action 和 category 的时候启动该 Activity -->
                <action
android:name="com.itheima.activitySkip.SecondActivity"/>
                <category
android:name="android.intent.category.DEFAULT"/>
                <!-- 设置数据协议 -->
                <data android:scheme="money"/>
                <!-- 配置数据的 mimeType:必须为 xxx/xxx 的格式，否则会报异常
-->
                <data android:mimeType="data/mymime"/>
            </intent-filter>
            </activity>
```

2 在 FirstActivity 布局中添加一个按钮，给该按钮绑定事件，点击该按钮实现跳转到 SecondActivity 界面。该事件核心方法清单如下：

```
// 发送数据给 SecondActivity
    public void sendData2Second(View view) {
        // 创建一个 Intent 对象
        Intent intent = new Intent();
        // 设置 Action
        intent.setAction("com.itheima.activitySkip.SecondActivity");
        // 对于 android.intent.category.DEFAULT 类型的信息为 Android 系统
默认的信息，省略也可以
        intent.addCategory("android.intent.category.DEFAULT");
        intent.setDataAndType(Uri.parse("money:转账 100 元。"),
"data/mymime");
        //给 Intent 设置数据
        intent.putExtra("name", "张三");
        ArrayList<String> list = new ArrayList<String>();
        for(int i=0;i<10;i++){
            list.add("list"+(i+1));
        }
        //给你 Intent 设置字符串类型的集合
        intent.putStringArrayListExtra("list", list);
        //声明一个 Bundle 对象
        Bundle bundle = new Bundle();
        //在 Bundle 对象中绑定数据
        bundle.putString("pwd", "123456");
        //给 Intent 设置 Bundle 对象
        intent.putExtras(bundle);
        // 启动 Activity
        startActivity(intent);

    }
```

3 编写 SecondActivity 类，使其接收数据。

```
public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        //获取从当前 Context 对象中获取 Intent
        Intent intent = getIntent();
        //获取数据
        String data = intent.getData().toString();
        System.out.println(data);
        //从 Intent 中获取 Bundle 对象
        Bundle extras = intent.getExtras();
        //从 Bundle 中获取可以为 name 的数据
        String name = (String) extras.get("name");
        System.out.println("name="+name);
        //从 Bundle 对象中获取 key 为 pwd 的数据
        String pwd = (String) extras.get("pwd");
        System.out.println(pwd);
        //从 Bundle 中获取 ley 为 list 的数据
        List<String> list = (List<String>) extras.get("list");
        System.out.println(list);
    }
```

4 运行该工程，发现成功跳转到了 SecondActivity 界面，同时控制台也成功打印出了通过 Intent 传递数据。如下图：

Tag	Text
System.out	money:转账100元。
System.out	name=张三
System.out	123456
System.out	[list1, list2, list3, list4, list5, list6, list7, list8, list9, list10]

5.案例-人品计算器（★★）

知识点：

1．跳转时使用 startActivityForResult 方法开启新页面

2．在被开启的新的页面里，调用 setResult 方法设置回传的数据.

注意：设置完后不会立刻传递到前一个页面,而是等待当前页面关闭后，才会回传数据。

3. 回传数据给传递给前一个页面的 onActivityResult 方法，在此方法中对数据进行处理即可。

需求：在 MainActivity 页面中输入用户名，点击计算跳转到计算页面，计算页面计算完毕后将计算结果返回给 MainActivity 界面，该界面将计算结果显示出来。

1 创建一个新工程《人品计算器》，使用默认的 Activity 和默认的布局文件，修改布局文件。activity_main.xml 布局清单如下：

```
<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:background="@drawable/rp"
    android:orientation="vertical"
    tools:context=".MainActivity" >
    <EditText
        android:layout_centerVertical="true"
        android:hint="请输入姓名"
        android:id="@+id/et"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:layout_below="@id/et"
        android:layout_alignParentRight="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="计算"
        android:id="@+id/bt"
        android:onClick="click" />
    <TextView
        android:id="@+id/tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/bt"
        android:visibility="invisible"
        android:textSize="28sp"
        android:textColor="#0000ff"
        />
</RelativeLayout>
```

Tips：上面布局文件采用相对布局，其中 android:background="@drawable/rp" 属性是给界面设置背景图片，因此需要我们将图片添加到/res/drawable-hdpi 目录下面。

android:visibility="invisible"属性是设置该控件不显示,显示的时候我们可以通过代码来实现其显示。

2 编写 MainActivity 核心代码类

```
public class MainActivity extends Activity {
    private TextView tv;
    private EditText et;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView) findViewById(R.id.tv);
        et = (EditText) findViewById(R.id.et);
    }
    public void click(View view) {
        String data = et.getText().toString();
        if (data == null || "".equals(data.trim()) || data.length() >
10) {
            Toast.makeText(this, "大哥，您输入的是人名吗？",
Toast.LENGTH_SHORT).show();
            et.setText("");
            tv.setVisibility(View.INVISIBLE);
            return;
        }
        byte[] bytes = data.getBytes();
        int result = 0;
        for (byte b : bytes) {
            result += b;
        }
        result = Math.abs(result % 101);
        //创建一个新 Intent
        Intent intent = new Intent(this,CalcActivity.class);
        //将数据绑定到 Intent 中
        intent.putExtra("rp", result);
        //打开一个新的 Activity，并接收器返回数据 请求码为 1，我们可以将请
求码理解为消息接应的暗号
        startActivityForResult(intent, 1);
```

```
    }
    /**
     * 当当前Activity用startActivityForResult方式调用另外一个Activity
的时候,
     * 另外一个 Activity 将数据返回后会 Android 系统会调用该方法
     *
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        //从当前 Intent 中获取数据
        Bundle extras = data.getExtras();
        String result  = (String) extras.get("result");
        //在也页面显示计算结果
        tv.setVisibility(View.VISIBLE);
        tv.setText(result);
    }

}
```

3 新建 CalcActivity 类继承 Activity 类，并在 AndroidManifest.xml 中添加该类。

```
public class CalcActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //获取当前 Intent
        Intent intent = getIntent();
        //获取 Intent 中绑定的数据
        Bundle extras = intent.getExtras();
        //获取 rp
        int data = (Integer) extras.get("rp");
        //计算人品结果
        String result = getRPText(data);
        //创建一个新的 Intent
        Intent intent2 = new Intent();
        //将结果绑定到 Intent 中
        intent2.putExtra("result", result);
        //返回请求码和结果
        setResult(1, intent2);
```

```
        //关闭当前 Activity
        finish();
    }
private String getRPText(int rp) {
        String rpText = null;
        if (rp == 0) {
            rpText = "你一定不是人吧？怎么一点人品都没有？！";
        } else if (rp > 0 && rp <= 5) {
            rpText = "算了，跟你没什么人品好谈的...";
        } else if (rp > 5 && rp <= 10) {
            rpText = "是我不好...不应该跟你谈人品问题的...";
        } else if (rp > 10 && rp <= 15) {
            rpText = "杀过人没有?放过火没有?你应该无恶不做吧?";
        } else if (rp > 15 && rp <= 20) {
            rpText = "你貌似应该三岁就偷看隔壁大妈洗澡的吧...";
        } else if (rp > 20 && rp <= 25) {
            rpText = "你的人品之低下实在让人惊讶啊...";
        } else if (rp > 25 && rp <= 30) {
            rpText = "你的人品太差了。你应该有干坏事的嗜好吧?";
        } else if (rp > 30 && rp <= 35) {
            rpText = "你的人品真差！肯定经常做偷鸡摸狗的事...";
        } else if (rp > 35 && rp <= 40) {
            rpText = "你拥有如此差的人品请经常祈求佛祖保佑你吧...";
        } else if (rp > 40 && rp <= 45) {
            rpText = "老实交待..那些论坛上面经常出现的偷拍照是不是你的杰
作?";
        } else if (rp > 45 && rp <= 50) {
            rpText = "你随地大小便之类的事没少干吧?";
        } else if (rp > 50 && rp <= 55) {
            rpText = "你的人品太差了..稍不小心就会去干坏事了吧?";
        } else if (rp > 55 && rp <= 60) {
            rpText = "你的人品很差了..要时刻克制住做坏事的冲动哦..";
        } else if (rp > 60 && rp <= 65) {
            rpText = "你的人品比较差了..要好好的约束自己啊..";
        } else if (rp > 65 && rp <= 70) {
            rpText = "你的人品勉勉强强..要自己好自为之..";
        } else if (rp > 70 && rp <= 75) {
            rpText = "有你这样的人品算是不错了..";
        } else if (rp > 75 && rp <= 80) {
            rpText = "你有较好的人品..继续保持..";
        } else if (rp > 80 && rp <= 85) {
```

```
            rpText = "你的人品不错..应该一表人才吧?";
        } else if (rp > 85 && rp <= 90) {
            rpText = "你的人品真好..做好事应该是你的爱好吧..";
        } else if (rp > 90 && rp <= 95) {
            rpText = "你的人品太好了..你就是当代活雷锋啊...";
        } else if (rp > 95 && rp <= 99) {
            rpText = "你是世人的榜样！";
        } else if (rp == 100) {
            rpText = "天啦！你不是人！你是神！！！";
        } else {
            rpText = "你的人品竟然负溢出了...我对你无语..";
        }
        return rpText;
    }
```

Tips：上面代码中的 getRPText 方法大家在练习的时候直接拷贝即可，没必要再重新写一遍。

运行上面的工程，启动后页面左侧图，输入名字后点击计算，结果如右侧图。

6.Activity 生命周期（★★★★）

Activity 有三种状态：

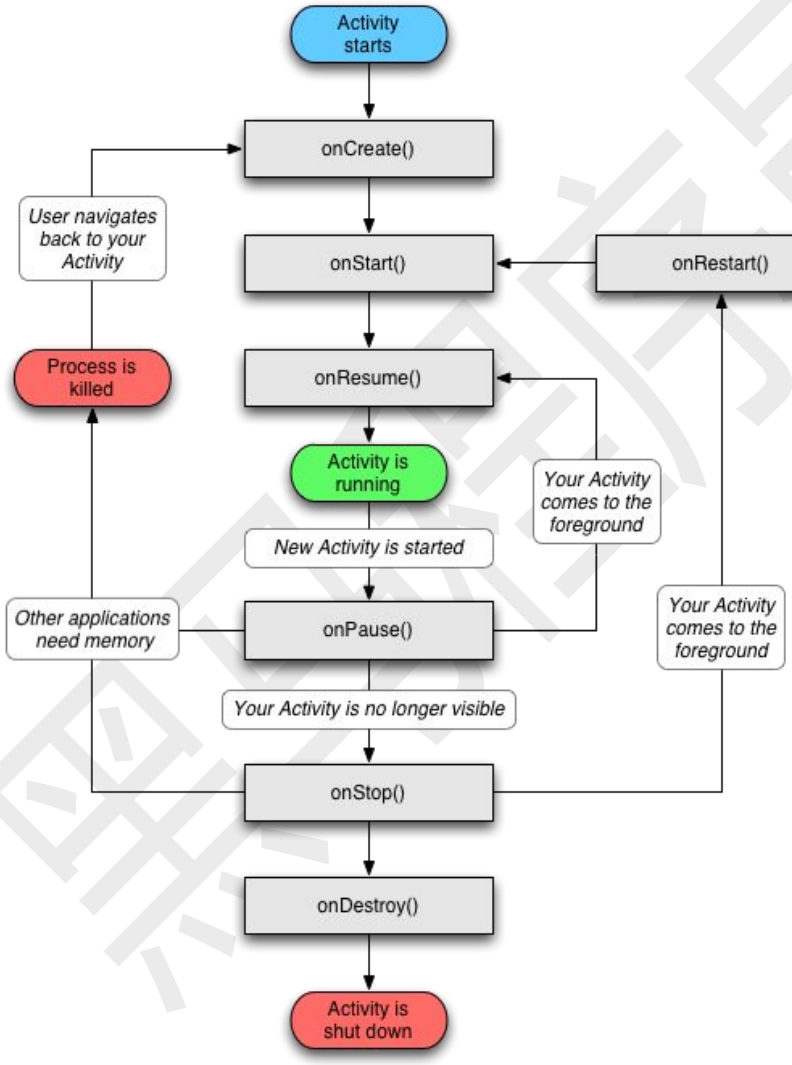
- 1、当它在屏幕前台时,响应用户操作的 Activity，它是激活或运行状态
- 2、当它上面有另外一个 Activity，使它失去了焦点但仍然对用户可见时，它处于暂停状态。
- 3、当它完全被另一个 Activity 覆盖时则处于停止状态。

当 Activity 从一种状态转变到另一种状态时，会调用以下保护方法来通知这种变化：

方法名	说明
void onCreate()	设置布局以及进行初始化操作
void onStart()	可见，但不可交互
void onRestart()	调用 onStart()
void onResume()	可见，可交互
void onPause()	部分可见，不可交互

void onStop()	完全不可见
void onDestroy()	销毁

Activity 生命周期图：



1、startActivity 开启一个 Activity 时，生命周期的过程是:

onCreate ->onStart(可见不可交互) ->onResume(可见可交互)

- 2、点击 back 键关闭一个 Activity 时，生命周期的过程是:

  onPause(部分可见不可交互)->onStop(完全不可见)->onDestroy(销毁)

- 3、当开启一个新的 Activity(以对话框形式)，新的 activity 把后面的 activity 给盖住一部分时，后面的 activity 的生命周期执行的方法是:

  onPause(部分可见，不可交互)

  注：指定 Activity 以对话框的形式显示，需在 activity 节点追加以下主题

  android:theme="@android:style/Theme.Dialog"

- 4、当把新开启的 Activity(以对话框形式)给关闭时，后面的 activity 的生命周期执行的方法是:

  onResume(可见，可交互)

- 5、当开启一个新的 activity 把后面的 activity 完全盖住时，生命周期的方法执行顺序是:

  onPause ->onStop(完全不可见)

- 6、当把新开启的 activity(完全盖住)给关闭时，生命周期的方法执行顺序是:

  onRestart ->onStart ->onResume(可见，可交互)

Tips：实际工作中常用的方法以及应用场景有：

onResume 可见，可交互.。把动态刷新的操作启动。

onPause 部分可见，不可交互. 把动态刷新的一些操作，给暂停了。

onCreate 初始化一些大量的数据。

onDestroy 把数据给释放掉，节省内存。

6.1 横竖屏切换时 Activity 的生命周期

横竖屏切换时,默认情况下会把 activity 先销毁再创建，在类似手机游戏这一类的应用中，这个体验是非常差的。不让 Activity 在横竖屏切换时销毁，只需要在清单文件声明 Activity 时配置<activity>节点的几个属性即可，其方式如下：

- 4.0 以下版本：

```
android:configChanges="orientation|keyboardHidden"
```

- 4.0 以上版本：

```
android:configChanges="orientation|screenSize"
```

- 兼容所有版本

```
android:configChanges="orientation|keyboardHidden|screenSize"
```

7.Activity 的任务栈（★★★★）

任务栈是用来提升用户体验而设计的:

(1)程序打开时就创建了一个任务栈，用于存储当前程序的 activity，所有的 activity 属于一个任务栈。

(2)一个任务栈包含了一个 activity 的集合，去有序的选择哪一个 activity 和用户进行交互:只有在任务栈栈顶的 activity 才可以跟用户进行交互。

(3)任务栈可以移动到后台，并且保留了每一个 activity 的状态. 并且有序的给用户列出它们

的任务，而且还不丢失它们状态信息。

(4)退出应用程序时：当把所有的任务栈中所有的 activity 清除出栈时,任务栈会被销毁,程序退出。

任务栈的缺点：

(1)每开启一次页面都会在任务栈中添加一个 Activity,而只有任务栈中的 Activity 全部清除出栈时，任务栈被销毁，程序才会退出,这样就造成了用，户体验差，需要点击多次返回才可以把程序退出了。

(2)每开启一次页面都会在任务栈中添加一个 Activity 还会造成数据冗余，重复数据太多，会导致内存溢出的问题(OOM)。

Tips：为了解决任务栈产生的问题，Android 为 Activity 设计了启动模式，那么下面的内容将介绍 Android 中 Activity 的启动模式，这也是最终要的内容之一。

8.Activity 的启动模式（★★★★）

启动模式（launchMode）在多个 Activity 跳转的过程中扮演着重要的角色，它可以决定是否生成新的 Activity 实例，是否重用已存在的 Activity 实例，是否和其他 Activity 实例公用一个 task 里。这里简单介绍一下 task 的概念，task 是一个具有栈结构的对象，一个 task 可以管理多个 Activity，启动一个应用，也就创建一个与之对应的 task。

Activity 一共有以下四种 launchMode：

- 1.standard

- 2.singleTop
- 3.singleTask
- 4.singleInstance

我们可以在 AndroidManifest.xml 配置<activity>的 android:launchMode 属性为以上四种之一即可。

下面我们结合实例一一介绍这四种 lanchMode：

8.1 standard

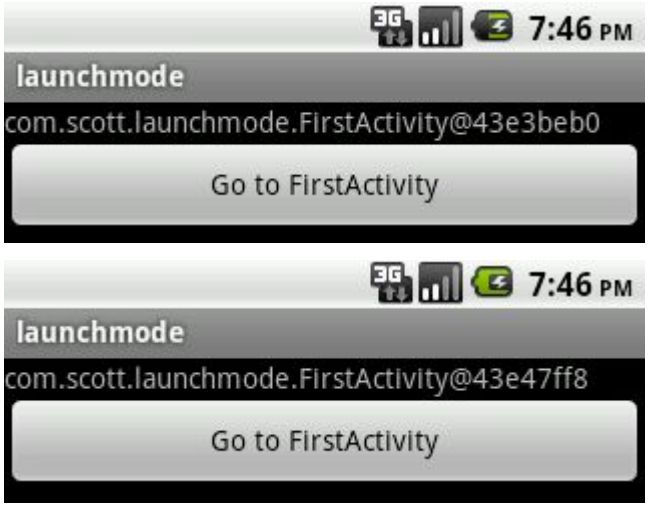
standard 模式是默认的启动模式，不用为<activity>配置 android:launchMode 属性即可，当然也可以指定值为 standard。

我们将创建一个 Activity，命名为 FirstActivity，来演示一下标准的启动模式。FirstActivity 代码如下：

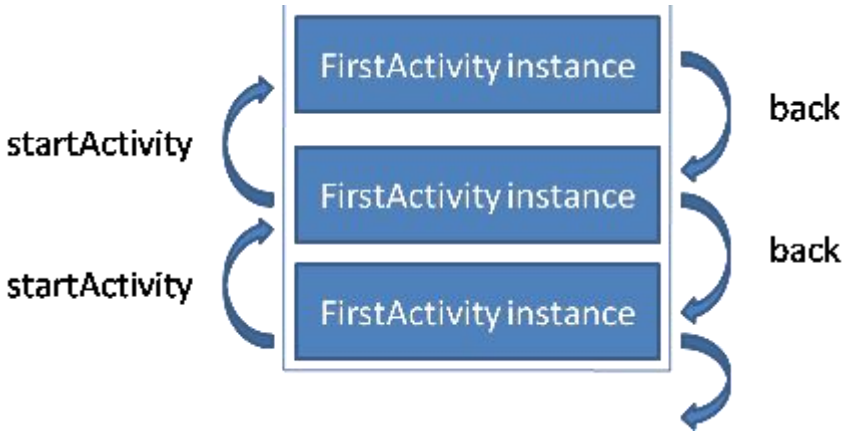
```
public class FirstActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first);
        TextView textView = (TextView) findViewById(R.id.tv);
        textView.setText(this.toString());
        Button button = (Button) findViewById(R.id.bt);
        button.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
             Intent intent = new Intent(FirstActivity.this,
FirstActivity.class);
             startActivity(intent);
          }
       });
    }
}
```

FirstActivity 界面中的 TextView 用于显示当前 Activity 实例的序列号，Button 用于跳转到下一个 FirstActivity 界面。

然后我们连续点击几次按钮，将会出现下面的现象：



我们注意到都是 FirstActivity 的实例，但序列号不同，并且我们需要连续按后退键两次，才能回到第一个 FirstActivity。standard 模式的原理如下图所示：

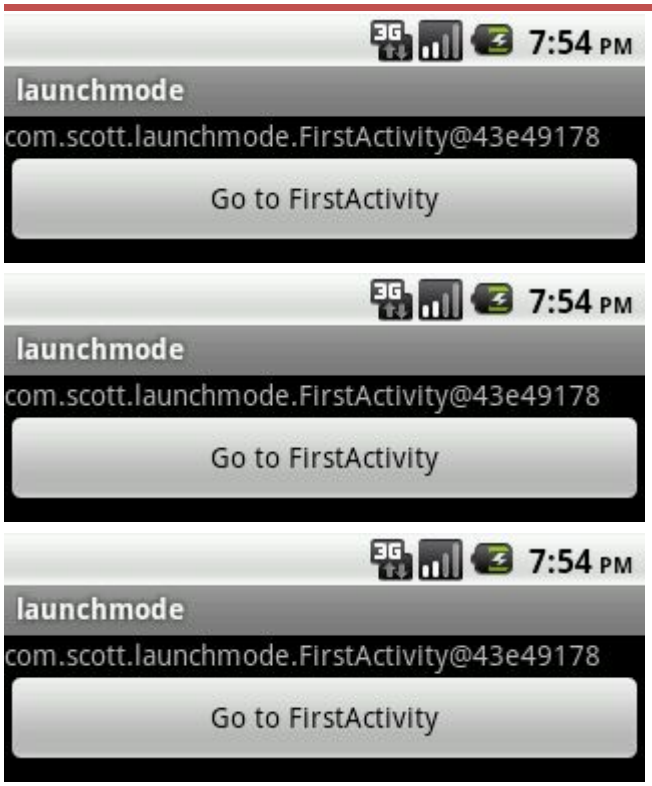


如图所示，每次跳转系统都会在 task 中生成一个新的 FirstActivity 实例，并且放于栈结构的顶部，当我们按下后退键时，才能看到原来的 FirstActivity 实例。

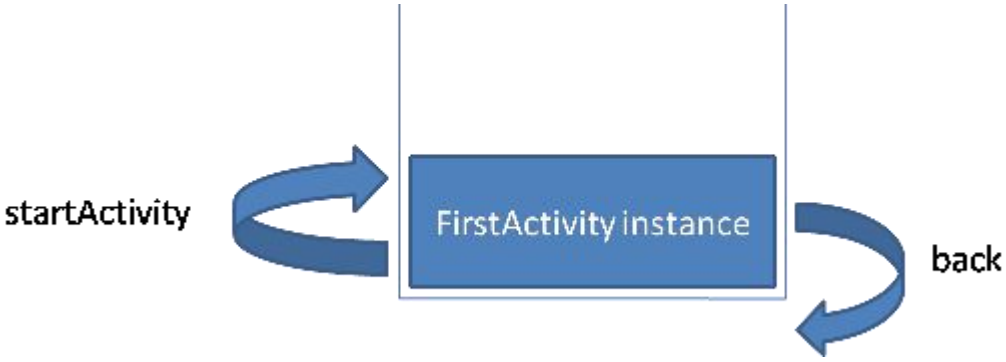
这就是 standard 启动模式，不管有没有已存在的实例，都生成新的实例。

8.2 singleTop

我们在上面的基础上为<activity>指定属性 **android:launchMode="singleTop"**，系统就会按照 singleTop 启动模式处理跳转行为。我们重复上面几个动作，将会出现下面的现象：



我们看到这个结果跟 standard 有所不同，三个序列号是相同的，也就是说使用的都是同一个 FirstActivity 实例；如果按一下后退键，程序立即退出，说明当前栈结构中只有一个 Activity 实例。singleTop 模式的原理如下图所示：



正如上图所示，跳转时系统会先在栈结构中寻找是否有一个 FirstActivity 实例正位于栈顶，如果有则不再生成新的，而是直接使用。也许朋友们会有疑问，我只看到栈内只有一个 Activity，如果是多个 Activity 怎么办，如果不是在栈顶会如何？我们接下来再通过一个示例来证实一下大家的疑问。

我们再新建一个 Activity 命名为 SecondActivity，如下：

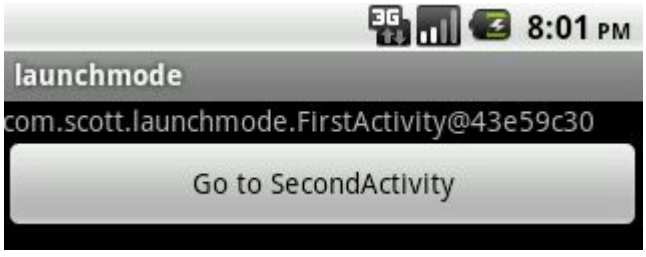
```
public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        TextView textView = (TextView) findViewById(R.id.tv);
         textView.setText(this.toString());
         Button button = (Button) findViewById(R.id.button);
         button.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
              Intent intent = new Intent(SecondActivity.this,
FirstActivity.class);
              startActivity(intent);
           }
         });
    }
}
```

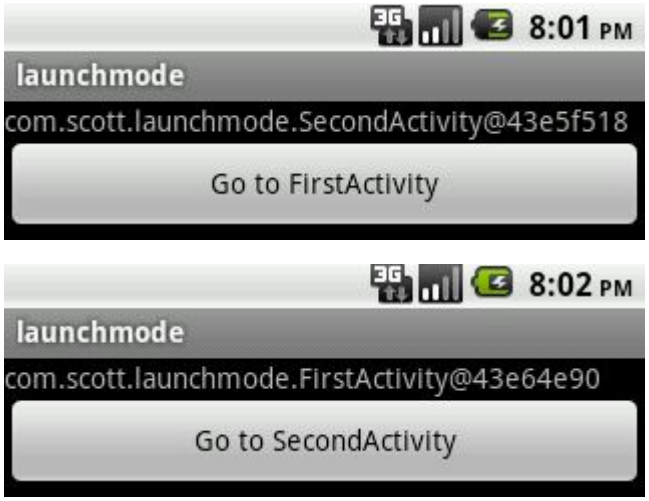
然后将之前的 FirstActivity 跳转代码改为：

```
Intent intent = new Intent(FirstActivity.this, SecondActivity.clas
s);
startActivity(intent);
```

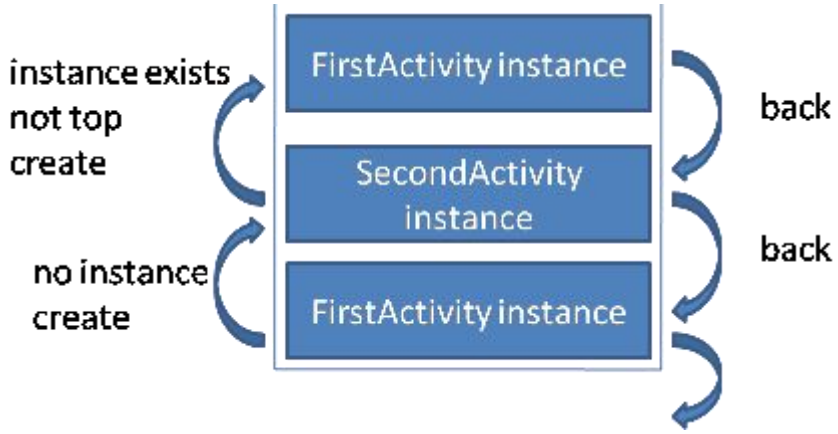
这时候，FirstActivity 会跳转到 SecondActivity，SecondActivity 又会跳转到 FirstActivity。

演示结果如下：





我们看到，两个 FirstActivity 的序列号是不同的，证明从 SecondActivity 跳转到 FirstActivity 时生成了新的 FirstActivity 实例。原理图如下：



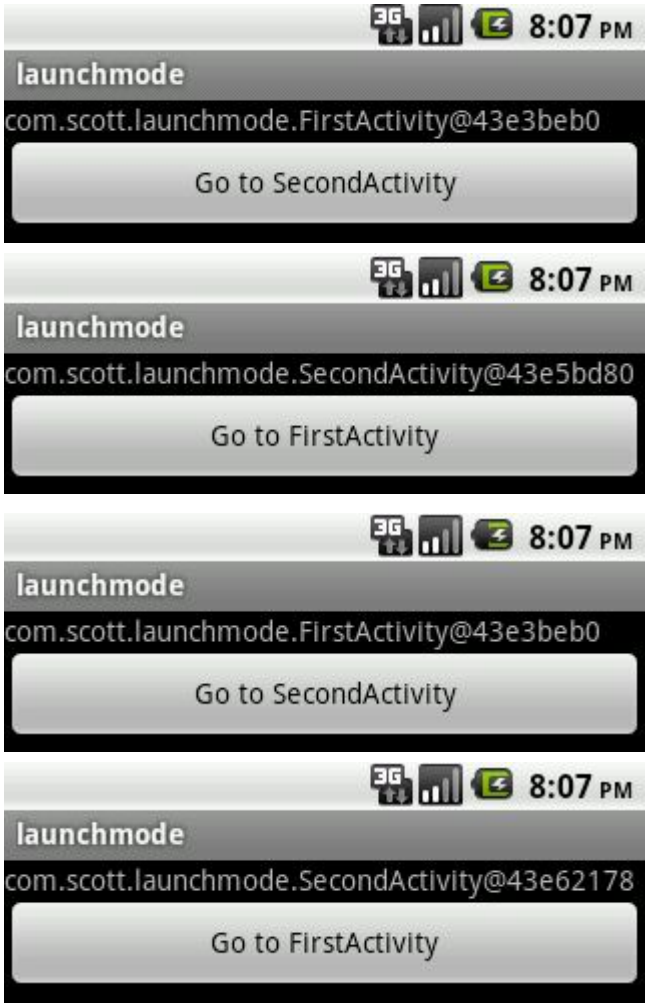
我们看到 ,当从 SecondActivity 跳转到 FirstActivity 时 ,系统发现存在有 FirstActivity 实例,但不是位于栈顶，于是重新生成一个实例。

这就是 singleTop 启动模式，如果发现有对应的 Activity 实例正位于栈顶，则重复利用，不再生成新的实例。

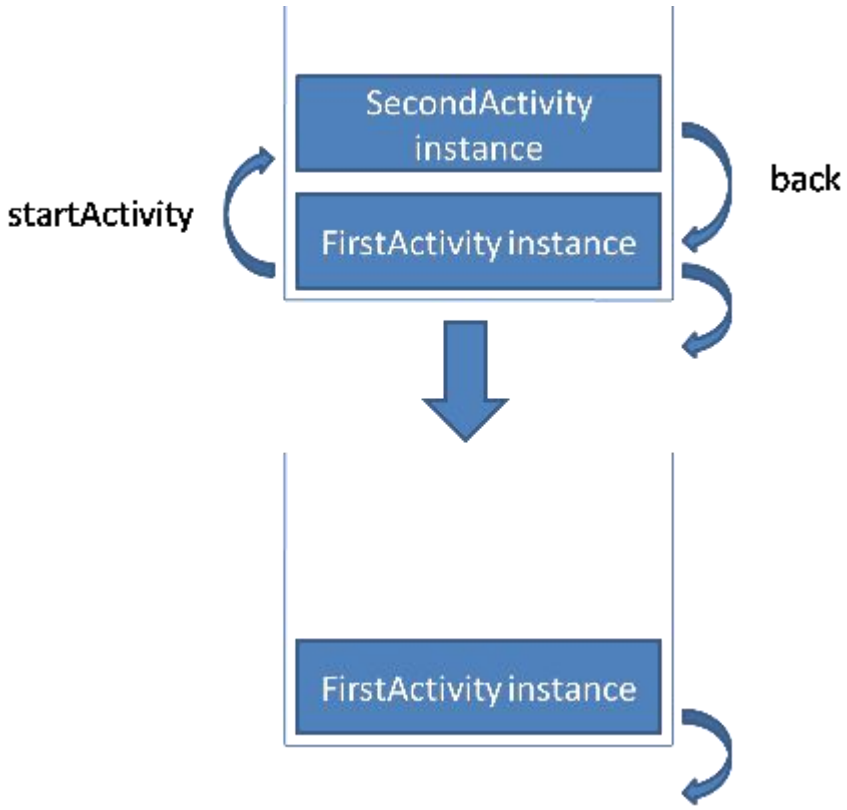
8.3 singleTask

在上面的基础上我们修改 FirstActivity 的属性

android:launchMode="singleTask"。演示的结果如下：



我们注意到，在上面的过程中，FirstActivity 的序列号是不变的，SecondActivity 的序列号却不是唯一的，说明从 SecondActivity 跳转到 FirstActivity 时，没有生成新的实例，但是从 FirstActivity 跳转到 SecondActivity 时生成了新的实例。singleTask 模式的原理图如下图所示：



在图中的下半部分是 SecondActivity 跳转到 FirstActivity 后的栈结构变化的结果，我们注意到，SecondActivity 消失了，没错，在这个跳转过程中系统发现有存在的 FirstActivity 实例，于是不再生成新的实例，而是将 FirstActivity 之上的 Activity 实例统统出栈，将 FirstActivity 变为栈顶对象，显示到幕前。也许朋友们有疑问，如果将 SecondActivity 也设置为 singleTask 模式，那么 SecondActivity 实例是不是可以唯一呢？在我们这个示例中是不可能的，因为每次从 SecondActivity 跳转到 FirstActivity 时，SecondActivity 实例都被迫出栈，下次等 FirstActivity 跳转到 SecondActivity 时，找不到存在的 SecondActivity 实例，于是必须生成新的实例。但是如果我们有 ThirdActivity，让 SecondActivity 和 ThirdActivity 互相跳转，那么 SecondActivity 实例就可以保证唯一。

这就是 singleTask 模式，如果发现有对应的 Activity 实例，则使此 Activity 实例之上的其他 Activity 实例统统出栈，使此 Activity 实例成为栈顶对象，显示到幕前。

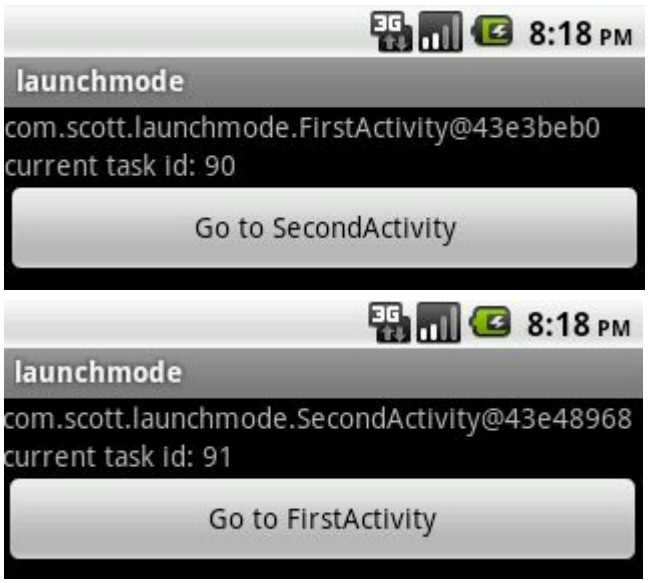
8.4 singleInstance

这种启动模式比较特殊，因为它会启用一个新的栈结构，将 Activity 放置于这个新的栈结构中，并保证不再有其他 Activity 实例进入。

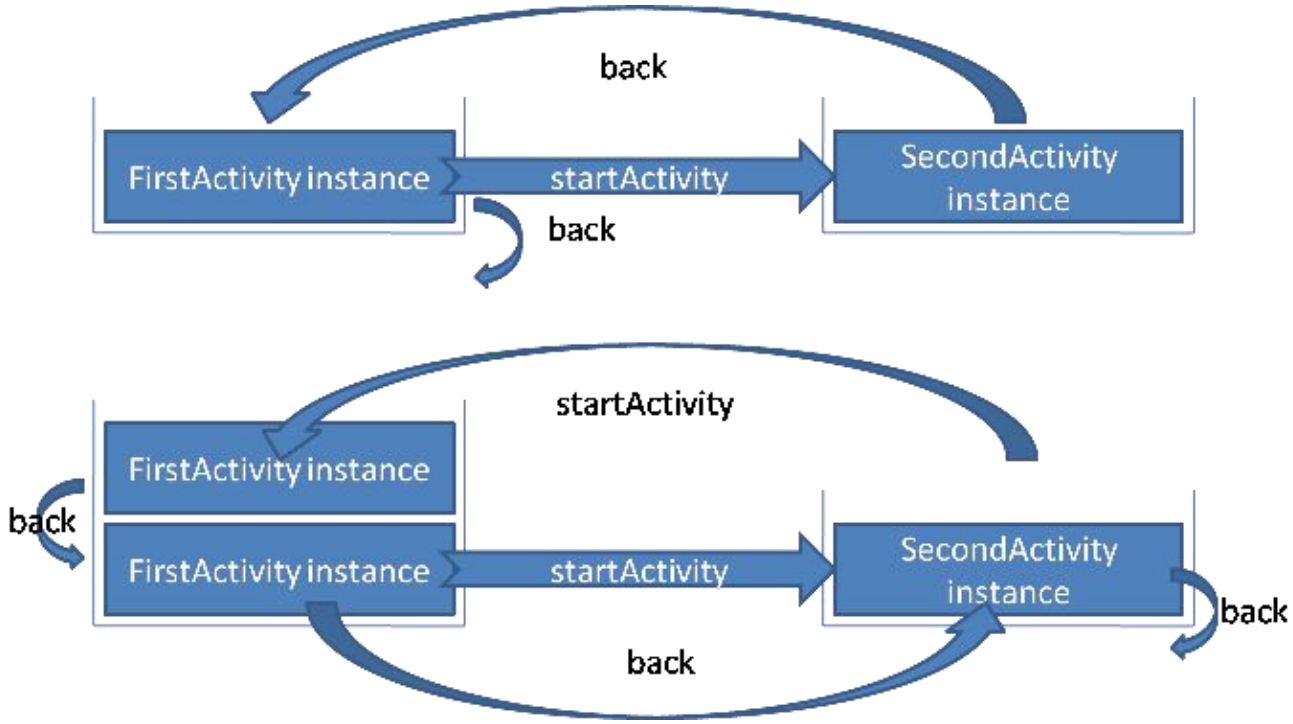
我们修改 FirstActivity 的 launchMode="standard"，SecondActivity 的 launchMode="singleInstance"，由于涉及到了多个栈结构，我们需要在每个 Activity 中显示当前栈结构的 id，所以我们为每个 Activity 添加如下代码：

```
TextView taskIdView = (TextView) findViewById(R.id.taskIdView);
taskIdView.setText("current task id: " + this.getTaskId());
```

然后我们再演示一下这个流程：

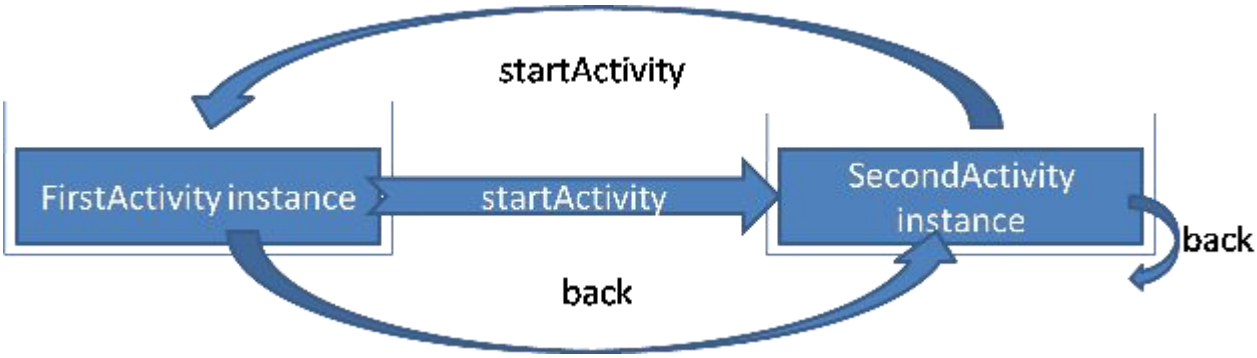


我们发现这两个 Activity 实例分别被放置在不同的栈结构中，关于 singleInstance 的原理图如下



我们看到从 FirstActivity 跳转到 SecondActivity 时，重新启用了一个新的栈结构，来放置 SecondActivity 实例，然后按下后退键，再次回到原始栈结构；图中下半部分显示的在 SecondActivity 中再次跳转到 FirstActivity，这个时候系统会在原始栈结构中生成一个 FirstActivity 实例，然后回退两次，注意，并没有退出，而是回到了 SecondActivity，为什么呢？是因为从 SecondActivity 跳转到 FirstActivity 的时候，我们的起点变成了 SecondActivity 实例所在的栈结构，这样一来，我们需要“回归”到这个栈结构。

如果我们修改 FirstActivity 的 launchMode 值为 singleTop、singleTask、singleInstance 中的任意一个，流程将会如图所示：



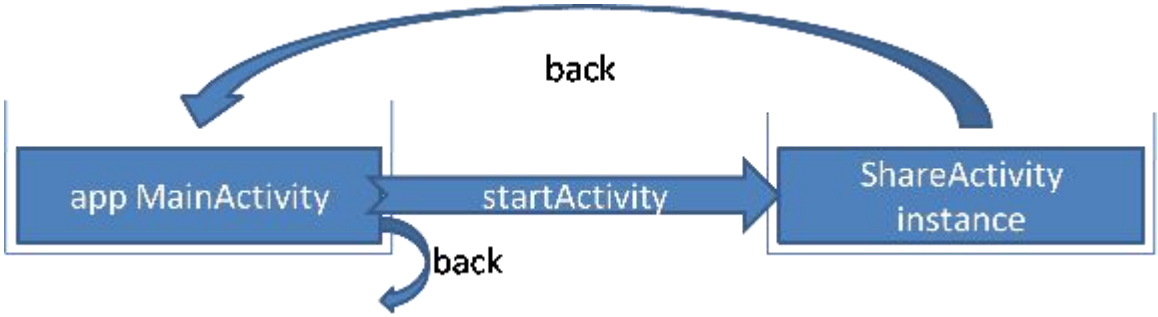
singleInstance 启动模式可能是最复杂的一种模式，为了帮助大家理解，我举一个例子，假如我们有一个 share 应用，其中的 ShareActivity 是入口 Activity，也是可供其他应用调用的 Activity，我们把这个 Activity 的启动模式设置为 singleInstance，然后在其他应用中调用。我们编辑 ShareActivity 的配置：

```
 <activity
   android:name=".ShareActivity"
   android:launchMode="singleInstance" >
  <intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
   <intent-filter>
<action android:name="android.intent.action.SINGLE_INSTANCE_SHARE"
/>
<category android:name="android.intent.category.DEFAULT" />
   </intent-filter>
</activity>
```

然后我们在其他应用中这样启动该 Activity：

```
Intent intent = new Intent("android.intent.action.SINGLE_INSTANCE_
SHARE");
startActivity(intent);
```

当我们打开 ShareActivity 后再按后退键回到原来界面时，ShareActivity 做为一个独立的个体存在，如果这时我们打开 share 应用，无需创建新的 ShareActivity 实例即可看到结果 ,因为系统会自动查找 ,存在则直接利用。大家可以在 ShareActivity 中打印一下 taskId，看看效果。关于这个过程，原理图如下：



至此，本文档全部完！