前面介绍了通过launchMode设置Activity的启动模式。本章接着介绍Activity的启动模式相关内容，讲解的内容是Intent与启动模式相关的Flag，以及android:taskAffinity的属性。

### **Intent与启动模式相关的Flag简介**

这里仅仅对几个常用的与启动模式相关的Flag进行介绍。

#### **FLAG\_ACTIVITY\_NEW\_TASK**

很少单独使用FLAG\_ACTIVITY\_NEW\_TASK，通常与FLAG\_ACTIVITY\_CLEAR\_TASK或FLAG\_ACTIVITY\_CLEAR\_TOP联合使用。因为单独使用该属性会导致奇怪的现象，通常达不到我们想要的效果！尽管如何，后面还是会通过"FLAG\_ACTIVITY\_NEW\_TASK示例一"和"FLAG\_ACTIVITY\_NEW\_TASK示例二"会向你展示单独使用它的效果。

#### **FLAG\_ACTIVITY\_SINGLE\_TOP**

在google的官方文档中介绍，它与launchMode="singleTop"具有相同的行为。实际上，的确如此！单独的使用FLAG\_ACTIVITY\_SINGLE\_TOP，就能达到和launchMode="singleTop"一样的效果。

#### **FLAG\_ACTIVITY\_CLEAR\_TOP**

顾名思义，FLAG\_ACTIVITY\_CLEAR\_TOP的作用清除"包含Activity的task"中位于该Activity实例之上的其他Activity实例。FLAG\_ACTIVITY\_CLEAR\_TOP和FLAG\_ACTIVITY\_NEW\_TASK两者同时使用，就能达到和launchMode="singleTask"一样的效果！

#### **FLAG\_ACTIVITY\_CLEAR\_TASK**

FLAG\_ACTIVITY\_CLEAR\_TASK的作用包含Activity的task。使用FLAG\_ACTIVITY\_CLEAR\_TASK时，通常会包含FLAG\_ACTIVITY\_NEW\_TASK。这样做的目的是启动Activity时，清除之前已经存在的Activity实例所在的task；这自然也就清除了之前存在的Activity实例！ 注意：当同时使用launchMode和上面的FLAG\_ACTIVITY\_NEW\_TASK等标签时，以FLAG\_ACTIVITY\_NEW\_TASK为标准。也就是说，代码的优先级比manifest中配置文件的优先级更高！ 下面，通过几个实例加深对这几个标记的理解。

#### **FLAG\_ACTIVITY\_NEW\_TASK标签**

#### **配置形式：**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.open.android.task5">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN"/>  
  
 <category android:name="android.intent.category.LAUNCHER"/>  
 </intent-filter>  
 </activity>  
 <activity android:name="SecondActivity" />  
 </application>  
  
</manifest>

#### **说明：**

在该实例中，有两个MainActivity和SecondActivity。

#### **使用案例：**

public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 findViewById(R.id.button).  
 setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(MainActivity.this, SecondActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  
 startActivity(intent);  
 }  
 });  
 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
 }  
 @Override  
 protected void onNewIntent(Intent intent) {  
 super.onNewIntent(intent);  
  
 Log.i("maweiqi", "onNewIntent：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
  
 }  
}

注意，跳转的Intent添加了FLAG\_ACTIVITY\_NEW\_TASK标志。

public class SecondActivity extends AppCompatActivity {  
  
 @Override  
 public void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_second);  
 Button button = (Button) findViewById(R.id.button2);  
 button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(SecondActivity.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
 }  
  
  
  
 @Override  
 protected void onNewIntent(Intent intent) {  
 super.onNewIntent(intent);  
  
 Log.i("maweiqi", "onNewIntent：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
  
 }  
}

#### **测试方式：**

MainActivity--> SecondActivity --> MainActivity--> SecondActivity

#### **输出的日志如下：**

onCreate：MainActivity TaskId: 73 hasCode:195694529  
onCreate：SecondActivity TaskId: 73 hasCode:54326677  
onCreate：MainActivity TaskId: 73 hasCode:121740648  
onCreate：SecondActivity TaskId: 73 hasCode:5761837

#### **在命令行中执行以下命令 adb shell dumpsys activity ,有以下输出：**

Running activities (most recent first):  
 TaskRecord{e3d17e2 #73 A=com.open.android.task5 U=0 sz=4}  
 Run #3: ActivityRecord{a34fa00 u0 com.open.android.task5/.SecondActivity t73}  
 Run #2: ActivityRecord{21172da u0 com.open.android.task5/.MainActivity t73}  
 Run #1: ActivityRecord{b1e1e64 u0 com.open.android.task5/.SecondActivity t73}  
 Run #0: ActivityRecord{f015a6e u0 com.open.android.task5/.MainActivity t73}  
  
 mResumedActivity: ActivityRecord{a34fa00 u0 com.open.android.task5/.SecondActivity t73}

#### **测试结果：**

1）全部在同一个task中

2）全部创建不同的实例

那如果MainActivity跳转去掉FLAG\_ACTIVITY\_NEW\_TASK？在SecondActivity清单文件添加singleTask，代码和流程跟之前一样，只贴出清单文件配置

#### **配置如下：**

<activity android:name="SecondActivity"  
 android:launchMode="singleTask"/>

#### **测试方式：**

MainActivity--> SecondActivity --> MainActivity--> SecondActivity

#### **输出的日志如下：**

onCreate：MainActivity TaskId: 74 hasCode:195694529  
onCreate：SecondActivity TaskId: 74 hasCode:54326677  
onCreate：MainActivity TaskId: 74 hasCode:38155814  
onNewIntent：SecondActivity TaskId: 74 hasCode:54326677

#### **在命令行中执行以下命令 adb shell dumpsys activity ,有以下输出：**

Running activities (most recent first):  
 TaskRecord{46d0de2 #74 A=com.open.android.task5 U=0 sz=2}  
 Run #1: ActivityRecord{b31602f u0 com.open.android.task5/.SecondActivity t74}  
 Run #0: ActivityRecord{f88d9dc u0 com.open.android.task5/.MainActivity t74}  
  
 mResumedActivity: ActivityRecord{b31602f u0 com.open.android.task5/.SecondActivity t74}

#### **测试结果：**

1）FLAG\_ACTIVITY\_NEW\_TASK的作用和singleTask具有不同的效果

那如果两个Activity的android:taskAffinity不相同呢？此时会导致什么效果呢？下面，我们通过示例来看看效果。

#### **配置如下：**

<activity android:name="SecondActivity"  
 android:taskAffinity="com.maweiqi.second"/>

代码回到最初MainActivity还是添加Flag标记，其他省略，如下：

public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 findViewById(R.id.button).  
 setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(MainActivity.this, SecondActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  
 startActivity(intent);  
 }  
 });  
 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
 }  
 @Override  
 protected void onNewIntent(Intent intent) {  
 super.onNewIntent(intent);  
  
 Log.i("maweiqi", "onNewIntent：" + getClass().getSimpleName() +  
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());  
  
 }  
}

#### **测试方式：**

MainActivity--> SecondActivity --> MainActivity--> SecondActivity

#### **输出的日志如下：**

onCreate：MainActivity TaskId: 77 hasCode:195694529  
onCreate：SecondActivity TaskId: 78 hasCode:54326677  
onCreate：MainActivity TaskId: 78 hasCode:38155814

#### **在命令行中执行以下命令 adb shell dumpsys activity ,有以下输出：**

Running activities (most recent first):  
 TaskRecord{65dda1d #78 A=com.maweiqi.second U=0 sz=2}  
 Run #2: ActivityRecord{845c9ed u0 com.open.android.task5/.MainActivity t78}  
 Run #1: ActivityRecord{f813328 u0 com.open.android.task5/.SecondActivity t78}  
 TaskRecord{dd8e492 #77 A=com.open.android.task5 U=0 sz=1}  
 Run #0: ActivityRecord{7a85f99 u0 com.open.android.task5/.MainActivity t77}  
  
 mResumedActivity: ActivityRecord{845c9ed u0 com.open.android.task5/.MainActivity t78}

#### **测试结果：**

1）当第二次进入到MainActivity中，再企图从MainActivity中进入到SecondActivity时，没有产生任何效果，仍然停留在MainActivity中！即第二次MainActivity--> SecondActivity压根就没发生！

#### **结果分析：**

1）当相互跳转的两个Activity的android:taskAffinity不同时，添加FLAG\_ACTIVITY\_NEW\_TASK：第一次启动SecondActivity时，会新建一个task，并将SecondActivity添加到该task中。这与singleTask产生的效果是一样的！但是，当企图再次从MainActivity进入到SecondActivity时，却什么也没有发生！

2）为什么呢？是因为此时SecondActivity实例已经存在，但是它所在的task的栈顶是MainActivity；而单独的添加FLAG\_ACTIVITY\_NEW\_TASK又不会"删除task中位于SecondActivity之上的Activity实例"，所以就没有发生跳转！

#### **FLAG\_ACTIVITY\_CLEAR\_TOP。**

修改MainActivity里面的点击事件如下：

Intent intent = new Intent(MainActivity.this, SecondActivity.class);  
intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK   
 | Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);  
startActivity(intent);

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

#### **测试方式：**

MainActivity--> SecondActivity --> MainActivity--> SecondActivity

#### **日志输出**

onCreate：MainActivity TaskId: 80 hasCode:58656936  
onCreate：SecondActivity TaskId: 80 hasCode:212434303  
onCreate：MainActivity TaskId: 80 hasCode:121740648  
onCreate：SecondActivity TaskId: 80 hasCode:15009890

#### **测试结果：**

1） MainActivity和SecondActivity在同一个task中！

2） 两个SecondActivity是不同的实例。

#### **结果分析：**

这与没有添加FLAG\_ACTIVITY\_CLEAR\_TOP时效果一样！这说明，当相互跳转的两个Activity的android:taskAffinity一样时，不会产生任何效果！

接下来，看看不同android:taskAffinity的情况，代码一致，清单文件修改

#### **文件配置**

<activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN"/>  
  
 <category android:name="android.intent.category.LAUNCHER"/>  
 </intent-filter>  
 </activity>  
<activity android:name="SecondActivity"  
 android:taskAffinity="com.maweiqi.second"/>

#### **测试方式：**

MainActivity--> SecondActivity --> MainActivity--> SecondActivity

#### **日志输出**

onCreate：MainActivity TaskId: 83 hasCode:195694529  
onCreate：SecondActivity TaskId: 84 hasCode:54326677  
onCreate：MainActivity TaskId: 84 hasCode:190178689  
onCreate：SecondActivity TaskId: 84 hasCode:5761837

#### **在命令行中执行以下命令 adb shell dumpsys activity ,有以下输出：**

Running activities (most recent first):  
 TaskRecord{12e942 #84 A=com.maweiqi.second U=0 sz=1}  
 Run #1: ActivityRecord{e0674b5 u0 com.open.android.task5/.SecondActivity t84}  
 TaskRecord{62b9d53 #83 A=com.open.android.task5 U=0 sz=1}  
 Run #0: ActivityRecord{71ec08a u0 com.open.android.task5/.MainActivity t83}  
  
 mResumedActivity: ActivityRecord{e0674b5 u0 com.open.android.task5/.SecondActivity t84}

#### **测试结果：**

1）MainActivity和SecondActivity在不同task中！

2） 两个SecondActivity是不同实例。

#### **结果分析：**

FLAG\_ACTIVITY\_NEW\_TASK和FLAG\_ACTIVITY\_CLEAR\_TOP的使用和android:taskAffinity相关。 在同时使用FLAG\_ACTIVITY\_NEW\_TASK|FLAG\_ACTIVITY\_CLEAR\_TOP的情况下，以A启动B来说

1）当A和B的taskAffinity相同时：添加FLAG\_ACTIVITY\_NEW\_TASK|FLAG\_ACTIVITY\_CLEAR\_TOP没有任何作用。和没有添加时的效果一样！

2）当A和B的taskAffinity不同时：添加FLAG\_ACTIVITY\_NEW\_TASK|FLAG\_ACTIVITY\_CLEAR\_TOP后，如果该activity没有设置启动模式（即默认是standard），或者intent没有设置一个FLAG\_ACTIVITY\_SINGLE\_TOP 的flag，那么这个activity就会销毁，然后重新创建这个实例

### **总结：**

#### **Activity的启动模式**

在AndroidManifest.xml中，可以配置每个activity的启动模式：例如： android:launchMode="standard"

#### **（1） standard 标准模式**

此模式，不管有没有已存在的实例，都生成新的实例。每次调用startActivity()启动Activity时都会创建一个新的Activity放在栈顶，每次返回都会销毁实例并出栈，可以重复创建。

#### **（2） singletop 单一顶部模式**

如果任务栈的栈顶存在这个要开启的activity，不会重新创建新的activity，而是复用已存在的activity。保证栈顶如果存在，则不会重复创建，但如果不在栈顶，那么还是会创建新的实例。 应用场景：浏览器的书签

#### **（3） singletask 单一任务模式**

是一个比较严格的模式，在当前任务栈里面只能有一个实例存在，当开启activity的时候，就去检查在任务栈里面是否有实例已经存在，如果有实例存在就复用这个已经存在的activity，并且把这个activity上面的所有的别的activity都清空，复用这个已经存在的activity。 应用场景：BrowserActivity 浏览器界面 如果一个activity的创建需要占用大量的系统资源（cpu，内存）一般配置这个activity为singletask的启动模式。webkit内核(c) 初始化需要大量内存 js解析引擎 html渲染引擎 http解析，下载…减少内存开销，cpu占用。播放器的播放Activity

#### **（4） singleInstance**

这种启动模式比较特殊，它会启用一个新的任务栈，activity会运行在自己的任务栈里，这个任务栈里面只有一个实例存在并且保证不再有其他Activity实例进入。在整个手机操作系统里面只有一个实例存在。 应用场景：来电页面。InCallScreenActivity

### **最后的总结：**

实话说，关于任务栈的bug有点多，如果真的添加了启动模式，一定要多多测试，有的地方和官方文档描述完全不相符。 stackoverflow上也有人吐糟。 <http://stackoverflow.com/questions/9772927/flag-activity-new-task-clarification-needed>