### 源码分析相关面试题

* [Volley源码分析](http://www.jianshu.com/p/ec3dc92df581)
* [注解框架实现原理](http://www.jianshu.com/p/20da6d6389e1)
* [okhttp3.0源码分析](http://www.jianshu.com/p/9ed2c2f2a52c)
* [onSaveInstanceState源码分析](http://www.jianshu.com/p/cbf9c3557d64)
* [静默安装和源码编译](http://www.jianshu.com/p/2211a5b3c37f)

### Activity相关面试题

* [保存Activity的状态](http://www.jianshu.com/p/cbf9c3557d64)
* [深刻剖析activity启动模式(一)](http://www.jianshu.com/p/b33fd8c550bf)
* [深刻剖析activity启动模式(二)](http://www.jianshu.com/p/e1ea9e542112)

### 与XMPP相关面试题

* [XMPP协议优缺点](http://www.jianshu.com/p/2c04ac3c526a)
* [极光消息推送原理](http://www.jianshu.com/p/d88dc66908cf)

### 与性能优化相关面试题

* [内存泄漏和内存溢出区别](http://www.jianshu.com/p/5dd645b05c76)
* [UI优化和线程池实现原理](http://www.jianshu.com/p/c22398f8587f)
* [代码优化](http://www.jianshu.com/p/ebd41eab90df)
* [内存性能分析](http://www.jianshu.com/p/2665c31b9c2f)
* [内存泄漏检测](http://www.jianshu.com/p/1514c7804a06)
* [App启动优化](http://www.jianshu.com/p/f0f73fefdd43)
* [与IPC机制相关面试题](http://www.jianshu.com/p/de4793a4c2d0)

### 与登录相关面试题

* [oauth认证协议原理](http://www.jianshu.com/p/2a6ecbf8d49d)
* [token产生的意义](http://www.jianshu.com/p/9b7ce2d6c195)
* [微信扫一扫实现原理](http://www.jianshu.com/p/a9d1f21bd5e0)

### 与开发相关面试题

* [迭代开发的时候如何向前兼容新旧接口](http://www.jianshu.com/p/cbecadec98de)
* [手把手教你如何解决as jar包冲突](http://www.jianshu.com/p/30fdc391289c)
* [context的原理分析](http://www.jianshu.com/p/2706c13a1769)
* [解决ViewPager.setCurrentItem中间很多页面切换方案](http://www.jianshu.com/p/38ab6d856b56)

### 与人事相关面试题

* [人事面试宝典](http://www.jianshu.com/p/d61b553ff8c9)

### **实例验证singleTask启动模式**

上篇文章将activity的四种启动模式就基本介绍完了。为了加深对启动模式的了解，下面会通过一个简单的例子进行验证。由以上的介绍可知，standard和singleTop这两种启动模式行为比较简单，所以在下面的例子中，通过taskAffinity会对singleTask和singleInstance着重介绍。

#### **验证启动singleTask模式的activity时是否会创建新的任务**

这个实例中有三个Activity,分别为：MainActivity，SecondActivity和ThirdActivity。

#### **配置形式：**

<activity android:label="@string/app\_name"
 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:launchMode="singleTask">
 <intent-filter >
 <action android:name="com.maweiqi.SecondActivity"/>
 <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
 </activity>

 <activity android:name=".ThirdActivity"
 android:label="@string/app\_name" >
 </activity>

#### **说明：**

MainActivity和ThirdActivity都是标准的启动模式，而SecondActivity的启动模式为singleTask。

#### **使用案例：**

public class MainActivity extends AppCompatActivity {
 private static final String ACTIVITY\_NAME = "MainActivity";
 private static final String LOG\_TAG = "maweiqi";

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

 findViewById(R.id.button1).setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
 Intent intent = new Intent(MainActivity.this, SecondActivity.class);

 startActivity(intent);
 }
 });

 int taskId = getTaskId();
 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() +
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());
 }
}

public class SecondActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_second);

 findViewById(R.id.button2).setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
 Intent intent = new Intent(SecondActivity.this, ThirdActivity.class);
 startActivity(intent);
 }
 });

 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() + " TaskId: "
 + getTaskId() + " hasCode:" + this.hashCode());

 }
}

public class ThirdActivity extends AppCompatActivity {
 @Override
 public void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 Log.i("maweiqi", "onCreate：" + getClass().getSimpleName() +
 " TaskId: " + getTaskId() + " hasCode:" + this.hashCode());
 }
}

#### **测试方式：**

MainActivity -->SecondActivity -->ThirdActivity。

#### **输出的日志如下：**

onCreate：MainActivity TaskId: 21 hasCode:199785693
onCreate：SecondActivity TaskId: 21 hasCode:171956091

#### **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{838c0b8 #21 A=com.open.android.task1 U=0 sz=2}
 Run #1: ActivityRecord{abf1b0a u0 com.open.android.task1/.SecondActivity t21}
 Run #0: ActivityRecord{4e579b u0 com.open.android.task1/.MainActivity t21}

#### **测试结果：**

1）MainActivity和SecondActivity是启动在同一个任务中

2）在SecondActivity增加一个taskAffinity属性，如下所示：

<activity android:name=".SecondActivity"
 android:launchMode="singleTask"
 android:taskAffinity="com.maweiqi.second">
 <intent-filter >
 <action android:name="com.maweiqi.SecondActivity"/>
 <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
 </activity>

#### **测试方式：**

MainActivity -->SecondActivity -->ThirdActivity。

#### **输出的日志如下：**

onCreate：MainActivity TaskId: 24 hasCode:199785693
onCreate：SecondActivity TaskId: 25 hasCode:171956091
onCreate：ThirdActivity TaskId: 25 hasCode:76684615

#### **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{844539b #25 A=com.maweiqi.second U=0 sz=2}
 Run #2: ActivityRecord{2eb5348 u0 com.open.android.task1/.ThirdActivity t25}
 Run #1: ActivityRecord{119f0df u0 com.open.android.task1/.SecondActivity t25}
 TaskRecord{b730338 #24 A=com.open.android.task1 U=0 sz=1}
 Run #0: ActivityRecord{2e05e2a u0 com.open.android.task1/.MainActivity t24}

#### **测试结果：**

1）MainActivity和SecondActivity运行在不同的任务中了

2）ThirdActivity和SecondActivity运行在同一个任务中

在这里便引出了manifest文件中的一个重要属性，taskAffinity。在官方文档中可以得到关于taskAffinity的以下信息

#### **taskAffinity**

1）taskAffinity表示当前activity具有亲和力的一个任务（翻译不是很准确，原句为The task that the activity has an affinity for.），大致可以这样理解，这个 taskAffinity表示一个任务，这个任务就是当前activity所在的任务。

2）在概念上，具有相同的affinity的activity（即设置了相同taskAffinity属性的activity）属于同一个任务。

3） 一个任务的affinity决定于这个任务的根activity（root activity）的taskAffinity。

4） 默认情况下，一个应用中的所有activity具有相同的taskAffinity，即应用程序的包名。我们可以通过设置不同的taskAffinity属性给应用中的activity分组，也可以把不同的 应用中的activity的taskAffinity设置成相同的值。

5）为一个activity的taskAffinity设置一个空字符串，表明这个activity不属于任何task。

#### **结果分析：**

1）这就可以解释上面示例中的现象了，由第4条可知，MainActivity和SecondActivity具有不同的taskAffinity，MainActivity的taskAffinity为com.open.android.task1，SecondActivity的taskAffinity为com.maweiqi.second。

2）当新启动的activity（SecondActivity）是以FLAG\_ACTIVITY\_NEW\_TASK标志启动时（只有singleTask模式才会保证“single in task”,只使用FLAG\_ACTIVITY\_NEW\_TASK并不保证“single in task”，或者说singleTask包含了FLAG\_ACTIVITY\_NEW\_TASK，反之却不成立），framework会检索是否已经存在了一个affinity为com.maweiqi.second的任务（即一个TaskRecord对象）

3）如果存在这样的一个任务，则检查在这个任务中是否已经有了一个SecondActivity的实例。

3-1）如果已经存在一个SecondActivity的实例，则会重用这个任务和任务中的SecondActivity实例，将这个任务调到前台，清除位于SecondActivity上面的所有Activity，显示SecondActivity，并调用SecondActivity的onNewIntent（）；

3-2）如果不存在一个SecondActivity的实例，会在这个任务中创建SecondActivity的实例，并调用onCreate()方法

3-3）这是在设置了singleTask模式的情况下会这样，在没有设置singleTask模式的情况下（即默认的standard模式）使用FLAG\_ACTIVITY\_NEW\_TASK，并不会清除位于SecondActivity上面的所有Activity，而是会在task的上面重新创建一个SecondActivity。也就是说此时task中有两个SecondActivity。

4）如果不存在这样的一个任务，会创建一个新的affinity为com.maweiqi.second的任务，并且将SecondActivity启动到这个新的任务中

上面讨论的是设置taskAffinity属性的情况,如果SecondActivity只设置启动模式为singleTask,而不设置taskAffinity,即三个Activity的taskAffinity相同,都为应用的包名，那么SecondActivity是不会开启一个新任务的，framework中的判定过程如下：

1）在MainActivity启动SecondActivity时,发现启动模式为singleTask,那么设定他的启动标志为FLAG\_ACTIVITY\_NEW\_TASK

2）然后获得SecondActivity的taskAffinity,即为包名com.open.android.task1检查是否已经存在一个affinity为com.open.android.task1的任务,这个任务是存在的,就是MainActivity所在的任务,这个任务是在启动MainActivity时开启的

3）既然已经存在这个任务,就检索在这个任务中是否存在一个SecondActivity的实例,发现不存在在这个已有的任务中启动一个SecondActivity的实例

为了作一个清楚的比较,列出SecondActivity启动模式设为singleTask,并且taskAffinity设为com.maweiqi.second时的启动过程

1）在MainActivity启动SecondActivity时，发现启动模式为singleTask，那么设定他的启动标志为FLAG\_ACTIVITY\_NEW\_TASK

2）然后获得SecondActivity的taskAffinity,即com.maweiqi.second检查是否已经存在一个affinity为com.maweiqi.second的任务,这个任务是不存在的创建一个新的affinity为com.maweiqi.second的任务,并且将SecondActivity启动到这个新的任务中

framework中对任务和activity的调度是很复杂的,尤其是把启动模式设为singleTask或者以FLAG\_ACTIVITY\_NEW\_TASK标志启动时.所以,在使用singleTask和FLAG\_ACTIVITY\_NEW\_TASK时,要仔细测试应用程序.

#### **实例验证将两个不同app中的不同的singleTask模式的Activity的taskAffinity设成相同**

官方文档中提到，可以把不同的 应用中的activity的taskAffinity设置成相同的值，这样的话这两个activity虽然不在同一应用中，却会在运行时分配到同一任务中，下面对此进行验证，在这里，会使用上面的示例,并创建一个新的示例AndroidTaskTest3。AndroidTaskTest3由两个activity组成，分别为MianActivity和OtherActivity，在MianActivity中点击按钮会启动OtherActivity,两个应用代码和布局一样，仅列出清单文件，两份清单文件如下：

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.open.android.task1">

 <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"
 android:launchMode="singleTask"
 android:taskAffinity="com.maweiqi.second">
 <intent-filter >
 <action android:name="com.maweiqi.SecondActivity"/>
 <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
 </activity>
 <activity android:name=".ThirdActivity"/>
 </application>
</manifest>

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.open.android.task3">

 <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=".OtherActivity"
 android:taskAffinity="com.maweiqi.second"
 android:launchMode="singleTask">
 </activity>
 </application>

</manifest>

可以看到OtherActivity和SecondActivity的启动模式都被设置为singleTask,并且taskAffinity属性被设置为com.maweiqi.second.现在将这两个应用安装在设备上。执行以下操作：

#### **测试方式：**

1）MainActivity --> SecondActivity

2）MainActivity --> OtherActivity

启动AndroidTaskTest应用，在它的MianActivity中点击按钮开启SecondActivity，由上面的介绍可知secondActivity是运行在一个新任务中的，这个任务就是com.maweiqi.second。

然后按Home键回到Launcher，启动AndroidTaskTest3，在启动AndroidTaskTest3的入口MianActivity时，会自动启动新的任务，那么现在一共有三个任务，AndroidTaskTest的MianActivity和SecondActivity分别占用一个任务，AndroidTaskTest3的MianActivity也占用一个任务。

在AndroidTaskTest3的MianActivity中点击按钮启动OtherActivity，那么这个OtherActivity是在哪个任务中呢？

#### **日志输出**

\*\*\*AndroidTaskTest应用测试日志输出\*\*\*
onCreate：MainActivity TaskId: 29 hasCode:193251508
onCreate：SecondActivity TaskId: 30 hasCode:171956091

\*\*\*AndroidTaskTest3应用测试日志输出\*\*\*
onCreate：MainActivity TaskId: 31 hasCode:199785693
onCreate：OtherActivity TaskId: 30 hasCode:171956091

下面执行adb shell dumpsys activity命令，发现有以下输出：

Task id #30
 TaskRecord{7f2f34a #30 A=com.maweiqi.second U=0 sz=2}
 Intent { flg=0x10000000 cmp=com.open.android.task1/.SecondActivity }
 Hist #1: ActivityRecord{a0f9ded u0 com.open.android.task3/.OtherActivity t30}
 Intent { flg=0x10400000 cmp=com.open.android.task3/.OtherActivity }
 ProcessRecord{12090b5 27543:com.open.android.task3/u0a62}
 Hist #0: ActivityRecord{1048af6 u0 com.open.android.task1/.SecondActivity t30}
 Intent { flg=0x10000000 cmp=com.open.android.task1/.SecondActivity }
 ProcessRecord{5bc013e 26035:com.open.android.task1/u0a59}

 Task id #31
 TaskRecord{dce52bb #31 A=com.open.android.task3 U=0 sz=1}
 Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 cmp=com.open.android.task3/.MainActivity }
 Hist #0: ActivityRecord{f9e58c5 u0 com.open.android.task3/.MainActivity t31}
 Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 cmp=com.open.android.task3/.MainActivity }
 ProcessRecord{12090b5 27543:com.open.android.task3/u0a62}

 Task id #29
 TaskRecord{5b063d8 #29 A=com.open.android.task1 U=0 sz=1}
 Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 cmp=com.open.android.task1/.MainActivity }
 Hist #0: ActivityRecord{689947d u0 com.open.android.task1/.MainActivity t29}
 Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10000000 cmp=com.open.android.task1/.MainActivity }
 ProcessRecord{5bc013e 26035:com.open.android.task1/u0a59}

Running activities (most recent first):
 TaskRecord{7f2f34a #30 A=com.maweiqi.second U=0 sz=2}
 Run #3: ActivityRecord{a0f9ded u0 com.open.android.task3/.OtherActivity t30}
 TaskRecord{dce52bb #31 A=com.open.android.task3 U=0 sz=1}
 Run #2: ActivityRecord{f9e58c5 u0 com.open.android.task3/.MainActivity t31}
 TaskRecord{7f2f34a #30 A=com.maweiqi.second U=0 sz=2}
 Run #1: ActivityRecord{1048af6 u0 com.open.android.task1/.SecondActivity t30}
 TaskRecord{5b063d8 #29 A=com.open.android.task1 U=0 sz=1}
 Run #0: ActivityRecord{689947d u0 com.open.android.task1/.MainActivity t29}

mResumedActivity: ActivityRecord{a0f9ded u0 com.open.android.task3/.OtherActivity t30}

#### **结果分析**

1）所以由此可见,AndroidTaskTest1的SecondActivity和AndroidTaskTest3的OtherActivity是在同一任务中的

2）由上面adb shell dumpsys activity命令的输出结果还可以看出，AndroidTaskTest1和AndroidTaskTest3这两个应用程序会开启两个进程，他们的所有组件分别运行在独立的进程中

3）AndroidTaskTest1所在进程的进程号为u0a59，AndroidTaskTest3所在进程的进程号为u0a62

4）com.maweiqi.second任务中的两个activity属于不同的应用，并且运行在不同的进程中，这也说明了一个问题：任务（Task）不仅可以跨应用（Application），还可以跨进程（Process）。

#### **实例验证singleTask的另一意义：在同一个任务中具有唯一性**

singleTask模式只意味着“可以在一个新的任务中开启”，至于是不是真的会在新任务中开启，在framework中还有其他条件的限制。由上面的介绍可知，这个条件为：是否已经存在了一个由他的taskAffinity属性指定的任务。这一点具有迷惑性，我们在看到singleTask这个单词的时候，会直观的想到它的本意：single in task。即，在同一个任务中，只会有一个该activity的实例。现在让我们进行验证：

为了验证这种情况，需要修改一下上面用到的AndroidTaskTest1示例。增加一个FourthActivity，并且MianActivity，SecondActivity，ThirdActivity和FourthActivity这四个activity都不设置taskAffinity属性，并且将SecondActivity启动模式设为singleTask，这样这四个activity会在同一个任务中开启。代码和软件界面就不列出了，只列出清单文件。

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.open.android.task1">

 <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"
 android:launchMode="singleTask"
 >
 </activity>
 <activity android:name=".ThirdActivity"/>
 <activity android:name=".FourthActivity"/>
 </application>

</manifest>

#### **测试方式：**

MianActivity --> SecondActivity --> ThirdActivity --> FourthActivity

#### **日志输出**

onCreate：MainActivity TaskId: 34 hasCode:199785693
onCreate：SecondActivity TaskId: 34 hasCode:171956091
onCreate：ThirdActivity TaskId: 34 hasCode:240438941
onCreate：FourthActivity TaskId: 34 hasCode:168147209

由此可见这四个activity都是在同一个任务中的。再次执行adb shell dumpsys activity命令加以验证：

Running activities (most recent first):
 TaskRecord{d114530 #34 A=com.open.android.task1 U=0 sz=4}
 Run #3: ActivityRecord{edae6a3 u0 com.open.android.task1/.FourthActivity t34}
 Run #2: ActivityRecord{f9339a5 u0 com.open.android.task1/.ThirdActivity t34}
 Run #1: ActivityRecord{1a30096 u0 com.open.android.task1/.SecondActivity t34}
 Run #0: ActivityRecord{7e96fa4 u0 com.open.android.task1/.MainActivity t34}

#### **测试结果：**

1）同样可以说明目前这四个activity都运行在affinity为com.open.android.task1的任务中，即栈中的状态为MainActivity --> SecondActivity --> ThirdActivity --> FourthActivity。

下面执行在FourthActivity中点击按钮启动SecondActivity的操作，注意，SecondActivity的启动模式为singleTask，那么现在栈中的情况如何呢？

#### **日志输出**

没有日志输出，说明没有调用onCreate方法

再次执行adb shell dumpsys activity命令，有以下输出：

Running activities (most recent first):
 TaskRecord{d114530 #34 A=com.open.android.task1 U=0 sz=2}
 Run #1: ActivityRecord{1a30096 u0 com.open.android.task1/.SecondActivity t34}
 Run #0: ActivityRecord{7e96fa4 u0 com.open.android.task1/.MainActivity t34}

#### **测试结果：**

1）这时栈中的状态为MainActivity --> SecondActivity。确实确保了在任务中是唯一的，并且清除了同一任务中它上面的所有Activity。

2）那么这个SecondActivity的实例是重用的上次已有的实例还是重新启动了一个实例呢？可以观察系统Log， 发现系统Log没有改变，还是上面的四条Log。打印Log的语句是在各个Activity中的onCreate方法中执行的，没有打印出新的Log，说明SecondActivity的onCreate的方法没有重新执行，也就是说是重用的上次已经启动的实例，而不是销毁重建。

#### **结果分析：**

1）经过上面的验证，可以得出如下的结论：在启动一个singleTask的Activity实例时，如果系统中已经存在这样一个实例，就会将这个实例调度到任务栈的栈顶，并清除它当前所在任务中位于它上面的所有的activity。

### **实例验证singleInstance的行为**

考谷歌官方文档，singleInstance的特点可以归结为以下三条：

1）以singleInstance模式启动的Activity具有全局唯一性，即整个系统中只会存在一个这样的实例

2）以singleInstance模式启动的Activity具有独占性，即它会独自占用一个任务，被他开启的任何activity都会运行在其他任务中（官方文档上的描述为，singleInstance模式的Activity不允许其他Activity和它共存在一个任务中）

3）被singleInstance模式的Activity开启的其他activity，能够开启一个新任务，但不一定开启新的任务，也可能在已有的一个任务中开启

下面对这三个特点分别验证，所使用的示例同样为AndroidTaskTest1，只不过会进行一些修改，下面列出它的清单文件：

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.open.android.task1">

 <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"
 android:launchMode="singleInstance"
 >
 <intent-filter>
 <action android:name="com.maweiqi.ACTION\_MY"/>
 <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
 </activity>

 <activity android:name=".ThirdActivity"/>
 <activity android:name=".FourthActivity"/>
 </application>

</manifest>

由上面的清单文件可以知道，该应用包括四个activity，分别为MianActivity，SecondActivity，ThirdActivity，FourthActivity，其中SecondActivity启动模式设置为singleInstance。MianActivity可以开启SecondActivity，SecondActivity可以开启ThirdActivity。 并且为了可以在其他应用中开启SecondActivity，为SecondActivity设置了一个IntentFilter，这样就可以在其他应用中使用隐式Intent开启SecondActivity。

#### **测试方式：**

MianActivity --> SecondActivity --> ThirdActivity

为了更好的验证singleInstance的全局唯一性，还需要其他一个应用，新建AndroidTaskTest4。AndroidTaskTest4只需要一个MianActivity，在MainActivity中点击按钮会开启AndroidTaskTest1应用中的SecondActivity。开启AndroidTaskTest1应用中的SecondActivity的代码如下：

public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_main);
 Button button = (Button) findViewById(R.id.button);
 button.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 Intent intent = new Intent();
 intent.setAction("com.maweiqi.ACTION\_MY");
 startActivity(intent);
 }
 });
 }
}

#### **下面开始验证第一个特点：以singleInstance模式启动的Activity具有全局唯一性，即整个系统中只会存在一个这样的实例**

执行如下操作：安装AndroidTaskTest1应用，点击MainActivity中的按钮，开启SecondActivity，可以看到如下log输出：

#### **日志输出**

onCreate：MainActivity TaskId: 35 hasCode:199785693
onCreate：SecondActivity TaskId: 36 hasCode:171956091

执行adb shell dumpsys activity命令，有以下输出：

 Running activities (most recent first):
 TaskRecord{2b68544 #36 A=com.open.android.task1 U=0 sz=1}
 Run #1: ActivityRecord{6d9f8c u0 com.open.android.task1/.SecondActivity t36}
 TaskRecord{ae9a62d #35 A=com.open.android.task1 U=0 sz=1}
 Run #0: ActivityRecord{d0429f5 u0 com.open.android.task1/.MainActivity t35}

以上可以说明，singleInstance模式的Activity总是会在新的任务中运行(前提是系统中还不存在这样的一个实例) 。

下面验证它的全局唯一性，执行以下操作：安装另一个应用AndroidTaskTest4，在开启的MainActivity中点击按钮开启AndroidTaskTest1应用中的SecondActivity。看到打印出一条新的日志：

{act=com.maweiqi.ACTION\_MY cmp=com.open.android.task1/.SecondActivity} from uid 10063 on display 0

执行adb shell dumpsys activity命令，有以下输出：

 Running activities (most recent first):
 TaskRecord{2b68544 #36 A=com.open.android.task1 U=0 sz=1}
 Run #2: ActivityRecord{6d9f8c u0 com.open.android.task1/.SecondActivity t36}
 TaskRecord{a7e9abd #37 A=com.open.android.task4 U=0 sz=1}
 Run #1: ActivityRecord{f50eec0 u0 com.open.android.task4/.MainActivity t37}
 TaskRecord{ae9a62d #35 A=com.open.android.task1 U=0 sz=1}
 Run #0: ActivityRecord{d0429f5 u0 com.open.android.task1/.MainActivity t35}

#### **测试结果：**

1）开启的SecondActivity就是上次创建的编号为6d9f8c的SecondActivity。

2）Log中没有再次输出关于SecondActivity的信息，说明SecondActivity并没有重新创建

#### **结果分析：**

以singleInstance模式启动的Activity在整个系统中是单例的，如果在启动这样的Activiyt时，已经存在了一个实例，那么会把它所在的任务调度到前台，重用这个实例。

#### **下面开始验证第二个特点：以singleInstance模式启动的Activity具有独占性，即它会独自占用一个任务，被他开启的任何activity都会运行在其他任务中**

重新安装AndroidTaskTest1应用，点击MainActivity中的按钮，开启SecondActivity，在SecondActivity中点击按钮，开启ThirdActivity。可以看到有如下Log输出：

#### **测试方式：**

MainActivity --> SecondActivity --> ThirdActivity

#### **日志输出：**

onCreate：MainActivity TaskId: 42 hasCode:199785693
onCreate：SecondActivity TaskId: 43 hasCode:171956091
onCreate：ThirdActivity TaskId: 42 hasCode:76684615

执行adb shell dumpsys activity命令，有以下输出：

 Running activities (most recent first):
 TaskRecord{ace0710 #42 A=com.open.android.task1 U=0 sz=2}
 Run #3: ActivityRecord{322319f u0 com.open.android.task1/.ThirdActivity t42}
 TaskRecord{abc9c0e #43 A=com.open.android.task1 U=0 sz=1}
 Run #2: ActivityRecord{903a842 u0 com.open.android.task1/.SecondActivity t43}
 TaskRecord{ace0710 #42 A=com.open.android.task1 U=0 sz=2}
 Run #1: ActivityRecord{e3c0ddf u0 com.open.android.task1/.MainActivity t42}

#### **测试结果：**

SecondActivity所在的任务为43，被SecondActivity启动的ThirdActivity所在的任务为42，这就说明以singleInstance模式启动的Activity具有独占性，即它会独自占用一个任务，被他开启的任何activity都会运行在其他任务中

#### **下面开始验证第三个特点：被singleInstance模式的Activity开启的其他activity，能够在新的任务中启动，但不一定开启新的任务，也可能在已有的一个任务中开启**

有上面对第二个特点的验证可以看到，被SecondActivity启动的ThirdActivity并没有运行在一个新开启的任务中，而是和MainActivity运行在了同一个已有的任务中，那么在什么情况下ThirdActivity才会启动一个新的任务呢？

现在对程序的清单文件做以下修改，为ThirdActivity增加一个属性taskAffinity：

#### **配置如下：**

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

重新安装AndroidTaskTest1应用，执行和上一步中同样的操作：点击MainActivity中的按钮，开启SecondActivity，在SecondActivity中点击按钮，开启ThirdActivity。可以看到有如下输出：

onCreate：MainActivity TaskId: 44 hasCode:199785693
onCreate：SecondActivity TaskId: 45 hasCode:171956091
onCreate：ThirdActivity TaskId: 46 hasCode:76684615

执行adb shell dumpsys activity命令，有以下输出：

Running activities (most recent first):
 TaskRecord{3088b56 #46 A=com.maweiqi.second U=0 sz=1}
 Run #2: ActivityRecord{9eff1ed u0 com.open.android.task1/.ThirdActivity t46}
 TaskRecord{f9bb7d7 #45 A=com.open.android.task1 U=0 sz=1}
 Run #1: ActivityRecord{16c7b28 u0 com.open.android.task1/.SecondActivity t45}
 TaskRecord{e9af8c4 #44 A=com.open.android.task1 U=0 sz=1}
 Run #0: ActivityRecord{5c2ed47 u0 com.open.android.task1/.MainActivity t44}

#### **测试结果：**

1）被SecondActivity启动的ThirdActivity启动在了一个新的任务中，即在启动ThirdActivity时创建了一个新任务。这就说明被singleInstance模式的Activity A在开启另一activity B时，能够开启一个新任务，但是是不是真的开启新任务，还要受其他条件的限制，这个条件是：当前系统中是不是已经有了一个activity B的taskAffinity属性指定的任务。

其实这种行为和singleTask启动时的情况相同。在Activity的启动模式设置为singleTask时，启动时系统会为它加上FLAG\_ACTIVITY\_NEW\_TASK标志，而被singleInstance模式的Activity开启的activity，启动时系统也会为它加上FLAG\_ACTIVITY\_NEW\_TASK标志，所以他们启动时的情况是相同的，上面再验证singleTask时已经阐述过，现在重新说明一下：

#### **结果分析：**

由于ThirdActivity是被启动模式为singleInstance类型的Activity（即SecondActivity）启动的，framework会为它它加上FLAG\_ACTIVITY\_NEW\_TASK标志，这时 framework会检索是否已经存在了一个affinity为com.maweiqi.second（即ThirdActivity的taskAffinity属性）的任务

1）如果存在这样的一个任务，则检查在这个任务中是否已经有了一个ThirdActivity的实例.

1-1）如果已经存在一个ThirdActivity的实例，则会重用这个任务和任务中的ThirdActivity实例，将这个任务调到前台，清除位于ThirdActivity上面的所有Activity，显示ThirdActivity，并调用ThirdActivity的onNewIntent（）。

1-2）如果不存在一个ThirdActivity的实例，会在这个任务中创建ThirdActivity的实例，并调用onCreate()方法

1-3）需要注意（1-1）有一种特殊情况，MainActivity, SecondActivity, ThirdActivity, FourthActivity 都不设置 taskAffinity.FourthActivity 设置为 singleInstance。测试 MainActivity -> SecondActivity -> ThirdActivity -> FourthActivity -> SecondActivity，从 FourthActivity 跳转到 SecondActivity, 是新开的一个 SecondActivity， 不会销毁 原 SecondActivity 上面的 ThirdActivity。

2）如果不存在这样的一个任务，会创建一个新的affinity为com.maweiqi.second的任务，并且将ThirdActivity启动到这个新的任务中.

如果ThirdActivity不设置taskAffinity，即ThirdActivity和MainActivity的taskAffinity相同，都为应用的包名，那么ThirdActivity是不会开启一个新任务的.

#### **framework中的判定过程如下：**

1）在SecondActivity启动ThirdActivity时，因为SecondActivity是singleInstance的，所以设定ThirdActivity的启动标志为FLAG\_ACTIVITY\_NEW\_TASK

2）然后获得ThirdActivity的taskAffinity,即为包名com.open.android.task1

3）检查是否已经存在一个affinity为com.open.android.task1的任务，这个任务是存在的，就是MainActivity所在的任务，这个任务是在启动MainActivity时开启的

4） 既然已经存在这个任务,就检索在这个任务中是否存在一个ThirdActivity的实例,发现不存在

5）在这个已有的任务中启动一个SecondActivity的实例

#### **为了作一个清楚的比较，列出ThirdActivity的taskAffinity属性设为com.maweiqi.second时的启动过程**

1）在SecondActivity启动ThirdActivity时，因为SecondActivity是singleInstance的，那么设定ThirdActivity的启动标志为FLAG\_ACTIVITY\_NEW\_TASK

2）然后获得ThirdActivity的taskAffinity，即为com.maweiqi.second

3）检查是否已经存在一个affinity为com.maweiqi.second的任务，这个任务是不存在的

4） 创建一个新的affinity为com.maweiqi.second的任务，并且将ThirdActivity启动到这个新的任务

到此singleInstance也介绍完了。

### 小结

由上述可知，Task是Android Framework中的一个概念，Task是由一系列相关的Activity组成的，是一组相关Activity的集合。Task是以栈的形式来管理的。

我们在操作软件的过程中，一定会涉及界面的跳转。其实在对界面进行跳转时，Android Framework既能在同一个任务中对Activity进行调度，也能以Task为单位进行整体调度。在启动模式为standard或singleTop时，一般是在同一个任务中对Activity进行调度，而在启动模式为singleTask或singleInstance是，一般会对Task进行整体调度。

#### **对Task进行整体调度包括以下操作：**

1）按Home键，将之前的任务切换到后台 2）长按Home键，会显示出最近执行过的任务列表 3）在Launcher或HomeScreen点击app图标，开启一个新任务，或者是将已有的任务调度到前台 4）启动singleTask模式的Activity时，会在系统中搜寻是否已经存在一个合适的任务，若存在，则会将这个任务调度到前台以重用这个任务。如果这个任务中已经存在一个要启动的Activity的实例，则清除这个实例之上的所有Activity，将这个实例显示给用户。如果这个已存在的任务中不存在一个要启动的Activity的实例，则在这个任务的顶端启动一个实例。若这个任务不存在，则会启动一个新的任务，在这个新的任务中启动这个singleTask模式的Activity的一个实例。 5）启动singleInstance的Activity时，会在系统中搜寻是否已经存在一个这个Activity的实例，如果存在，会将这个实例所在的任务调度到前台，重用这个Activity的实例（该任务中只有这一个Activity），如果不存在，会开启一个新任务，并在这个新任务中启动这个singleInstance模式的Activity的一个实例。