《移动终端开发技术》

电子教案

第四单元 Activity 的启动模式

所属专业（教研室）：计算机软件技术

制定人：陈媛媛

合作人：

制定时间：2018 年 2 月

日照职业技术学院

<table>
<tr><td>单元标题</td><td colspan="2">Activity 的启动模式</td><td>单元教学学时</td><td>2 课时</td></tr>
<tr><td></td><td></td><td></td><td>在整体设计中的位置</td><td>第 12 次</td></tr>
<tr><td>授课班级</td><td colspan="2"></td><td>上课地点</td><td>一体化教室</td></tr>
<tr><td>上课时间</td><td colspan="4">周　　月　　日第　　节</td></tr>
<tr><td rowspan="2">教学
目标</td><td>能力目标</td><td colspan="2">知识目标</td><td>素质目标</td></tr>
<tr><td>能够熟练使用四种启动模式启动一个活动。</td><td colspan="2">1、了解 Android 中的任务栈；
2、掌握 Activity 的四种启动模式；
3、理解 Activity 的四种启动模式的特点和区别。</td><td>1、养成积极主动学习意识；
2、养成勤于动手的习惯。</td></tr>
<tr><td>教学重点、
难点</td><td colspan="4">教学重点：Activity 的四种启动模式
教学难点：Activity 的四种启动模式</td></tr>
<tr><td>教学
方法</td><td colspan="4">采用反转课堂教学模式，课前学生学习微课了解知识点，课上采用教师引导、演示，学生分组练习、讨论等教学方法。
运用多媒体、AndroidStudio 开发环境、实训助手、教学平台等辅助授课。</td></tr>
<tr><td>课前需掌握
的知识点</td><td colspan="4">1、Android 系统中的任务栈，类似于一个容器，用于管理所有的 Activity 实例。在存放 Activity 时，满足“先进后出（First-In/Last-Out）”的原则。
2、Activity 的启动模式有四种，分别是 standard、singleTop、singleTask 和 singleInstance 。 在 AndroidManifest.xml 中 ， 通 过 <activity> 标 签 的 android:launchMode 属性可以设置启动模式。</td></tr>
<tr><td>教学任务
分解</td><td colspan="4">任务一、standard 标准模式
任务二、singleTop 启动模式
任务三、singleTask 启动模式
任务四、singleInstance 启动模式</td></tr>
<tr><td>教学
总结</td><td colspan="4"></td></tr>
</table>

教学过程

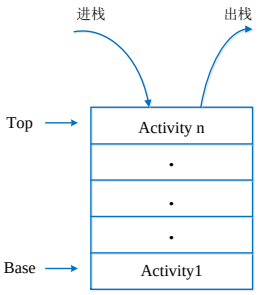
一、情景导入

1、引入任务栈（Task）

老师引导，大家想一下前面讲解 Activity 生命周期时，先后开启了两个 Activity，那么 Android 系统是如何管理这两个 Activity 的呢？

实际上，Android 系统采用任务栈（Task）的方式来管理 Activity 的实例，这个栈也称为返回栈（Back Stack）。当启动一个应用时，Android 就会为之创建一个任务栈。先启动的 Activity 压在栈底，后启动的 Activity 放在栈顶，通过启动模式可以控制 Activity 在任务栈中的加载方式。

Android 系统中的任务栈，类似于一个容器，用于管理所有的 Activity 实例。在存放 Activity 时，满足“先进后出（First-In/Last-Out）”的原则。 当我们按下 back 键或调用 finish（）方法去销毁一个活动时，出于栈顶的活动就会出栈，这时前一个入栈的活动就会重新处于栈顶的位置。系统总是会显示处于栈顶的活动给用户。



2、引出本单元相关学习内容

本节课我们将一起学习 standard、singleTop、singleTask 和 singleInstance 这四种启动模式的特点，并通过案例运用这四种模式启动一个活动。

二、课前测试

在在线教学平台上做以下练习：

1、（ ）启动模式与 standard 类似，不同的是，当启动的 Activity 已经位于栈顶时，则直接使用它不创建新的实例。

A. singleTask

B. singleTop

C. singleTask 和 singleTop

D. singleInstance

答案 B

2、Android 系统采用（）的方式来管理 Activity 的实例

A.任务栈（Task） B.生命周期

C.Intent D.Activity

答案 A

3、Activity 默认的启动模式是（）。

A. singleTask B. singleTop

C. standard D. singleInstance

答案 C

4、如果希望 Activity 在整个应用程序中只存在一个实例，可以使用（）模式

A. singleTask　B. singleTop

C. standard　D. singleInstance

答案 A

5、Android 系统中的任务栈，类似于一个容器，用于管理所有的 Activity 实例。在存放 Activity 时，满足“先进后出（First-In/Last-Out）”的原则（）。

答案 正确

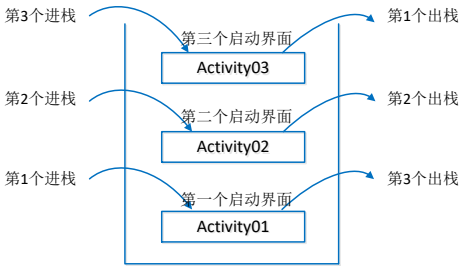
三、知识讲解

Android 是采用任务中的形式来管理 Activity 的。实际开发中，应根据特定的需求为每个 Activity 指定恰当的启动模式。

Activity 的启动模式有四种，分别是 standard、singleTop、singleTask 和 singleInstance。在 AndroidManifest.xml 中，通过<activity>标签的 android:launchMode 属性可以设置启动模式。

任务一、standard 标准模式

standard 是 Activity 默认的启动模式。在 standard 模式下，每当启动一个新的 Activity，它就会进入任务栈，并处于栈顶的位置，对于使用 standard 模式的 Activity，系统不会判断该 Activity 在栈中是否存在，每次启动都会创建一个新的实例。



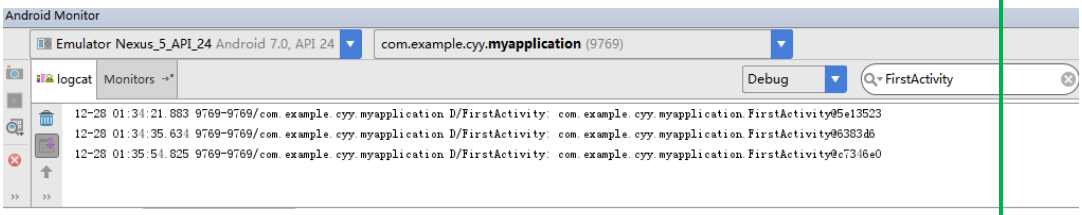
案例：关闭 AcitvityLifeCycleTest，使用 Intent 小节中的项目。

修改 FirstActivity

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("FirstActivity",this.toString());
    setContentView(R.layout.first_layout);
    Button button1=(Button) findViewById(R.id.button);
    button1.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View view) {
            Intent intent=new Intent(FirstActivity.this,FirstActivity.class);
            startActivity(intent);
        }
    });
}
```

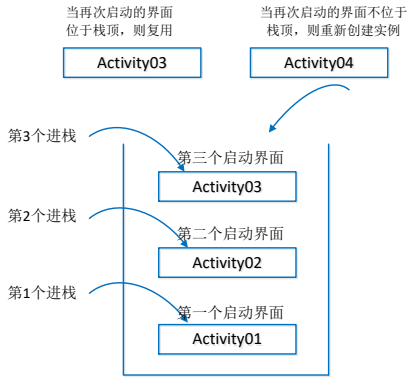
运行程序，查看结果：

点击两次按钮



任务二、singleTop 启动模式

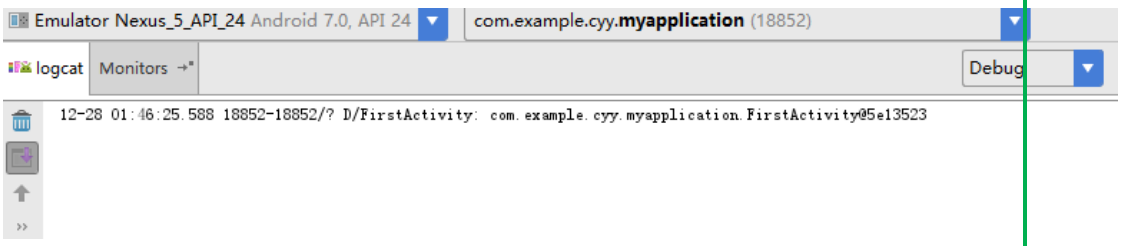
singleTop 启动模式与 standard 类似，不同的是，当启动的 Activity 已经位于栈顶时，则直接使用它不创建新的实例。如果启动的 Activity 没有位于栈顶时，则创建一个新的实例位于栈顶。



修改 AndroidManifest.xml，launchMode="singleTop"

```
<activity
  android:name=".FirstActivity"
  android:label="This is FirstActivity"
  android:launchMode="singleTop">
  <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

后再运行程序，无论点击几次按钮，结果为：



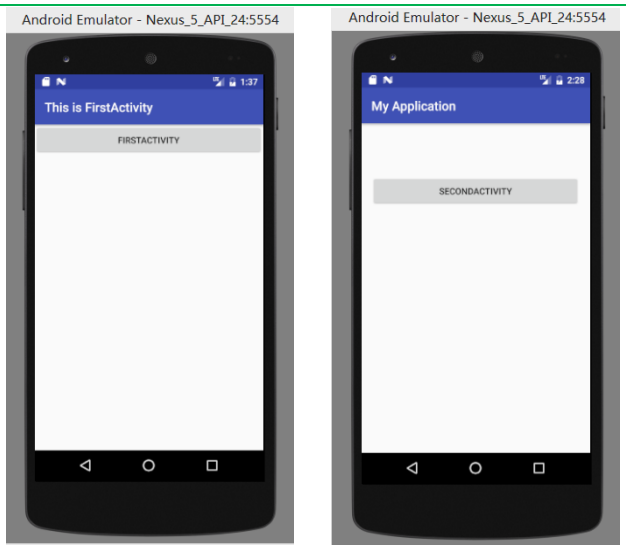
修改 FirstActivity

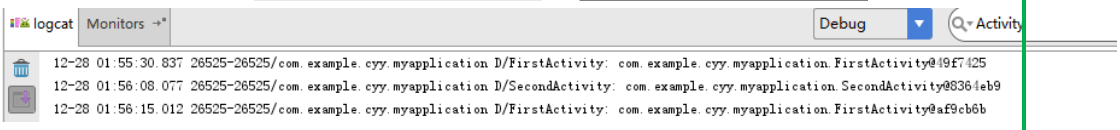
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("FirstActivity",this.toString());
    setContentView(R.layout.first_layout);
    Button button1=(Button) findViewById(R.id.button);
    button1.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View view) {
            Intent intent=new Intent(FirstActivity.this,SecondActivity.class);
            startActivity(intent);
        }
    });
}
```

修改 SecondActivity

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("SecondActivity",this.toString());
    setContentView(R.layout.second_layout);
    Button button2=(Button)findViewById(R.id.button2);
    button2.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View view) {
            Intent intent=new Intent(SecondActivity.this,FirstActivity.class);
            startActivity(intent);
             }
    });}
```

运行程序，查看结果：

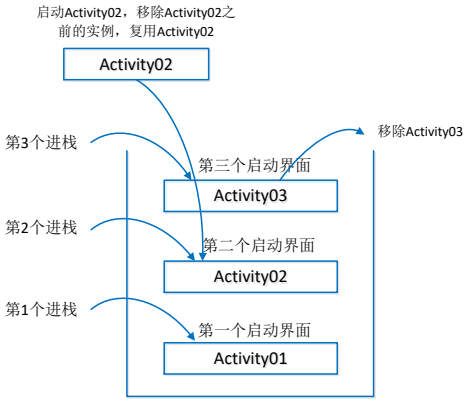




说明只有 FirstActivity 在栈顶时，不再创建实例，当 SecondActivity 进入到栈顶，再次点击按钮返回到 FirstActivity 时，重新创建实例。

任务三、singleTask 启动模式

如果希望 Activity 在整个应用程序中只存在一个实例，可以使用 singleTask 模式，当 Activity 的启动模式指定为 singleTask，每次启动该 Activity 时，系统首先会检查栈中是否存在该活动的实例，如果发现已经存在则直接使用该实例，并将当前 Activity 之上的所有 Activity 出栈，如果没有发现则创建一个新的实例。



案例：

修改 AndroidManifest.xml　FirstActivity 的 android:launchMode="singleTask"

修改 FirstActivity

加入如下代码：

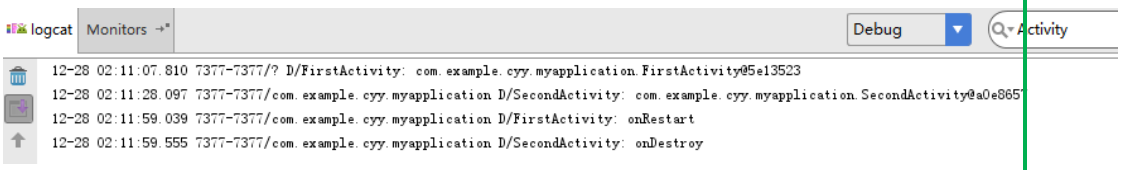
```
protected void onRestart(){
  super.onRestart();
  Log.d("FirstActivity","onRestart");
}
```

修改 SecondActivity

加入如下代码：

```
protected void onDestroy(){
  super.onDestroy();
  Log.d("SecondActivity","onDestroy");
}
```

查看运行结果：



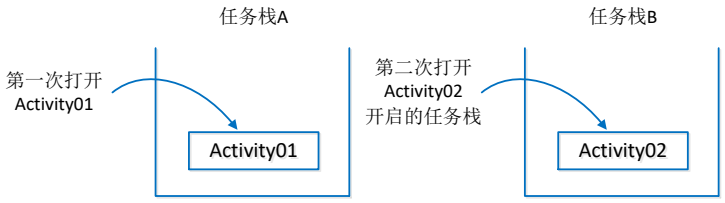
从打印的信息可以看出，在 SecondActivity 中启动 FirstActivity 时，会发现返回栈中已经存在一个 FirstActivity 实例，并且是在 SecondActivity 的下面，于是 SecondActivity 会从返回栈出栈，而 FirstActivity 从新成了栈顶活动，因此 FirstActivity 的 onRestart（）方法和 SecondActivity 的 onDestroy（）会执行。

任务四、singleInstance 模式

在程序开发中，如果需要 Activity 在整个系统中都只有一个实例，这时就需要用到 singleInstance 模式，不同于上述三种模式，指定为 singleInstance 模式的 Activity 会启动一个新的任务栈来管理这个 Activity。

singleInstance 模式加载 Activity 时，无论从哪个任务栈中启动该 Activity，只会创建一个 Activity 实例，并且会使用一个全新的任务栈来装载该 Activity 实例。采用这种模式启动 Activity 会分为以下两种情况，具体如下：

第一种：如果要启动的 Activity 不存在，系统会先创建一个新的任务栈，再创建该 Activity 的实例，并把该 Activity 加入栈顶。



第二种：如果要启动的 Activity 已经存在，无论位于哪个应用程序或者哪个任务栈中。系统都会把该 Activity 所在的任务栈转到前台，从而使该 Activity 显示出来。

案例：

修改 AndroidManifest.xml

```
<activity android:name=".SecondActivity"
  android:launchMode="singleInstance">
</activity>
```

修改 FirstActivity

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Log.d("FirstActivity","Task id is"+getTaskId());
```

```
    setContentView(R.layout.first_layout);
    Button button1=(Button) findViewById(R.id.button);
    button1.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            Intent intent=new Intent(FirstActivity.this,SecondActivity.class);
            startActivity(intent);
        }
    });
}
```

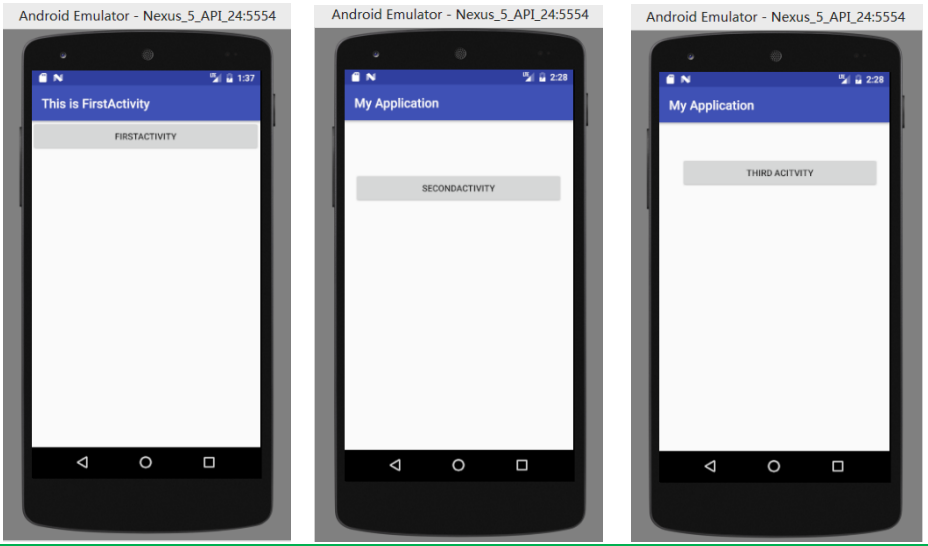
修改 SecondActivity

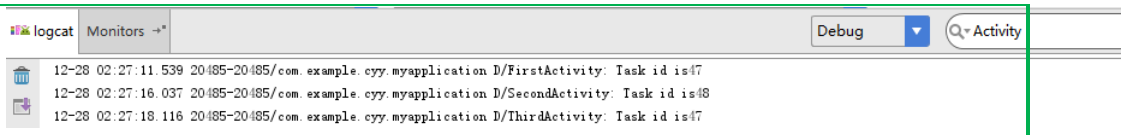
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("SecondActivity","Task id is"+getTaskId());
    setContentView(R.layout.second_layout);
    Button button2=(Button)findViewById(R.id.button2);
    button2.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View view) {
            Intent intent=new Intent(SecondActivity.this,ThirdActivity.class);
            startActivity(intent);
             }
    });}
```

新建 ThirdActivity

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d("ThirdActivity","Task id is"+getTaskId());
    setContentView(R.layout.third_layout);
}
```

运行查看结果：





可以看到 SecondActivity 的 Task id 不同于 FirstActivity 和 ThirdActivity，SecondActivity 确实存放在一个单独的返回栈里的，这个栈只有 SecondActivity 一个活动。

四、演示作品

各小组演示汇报作品，教师与其他小组总结评价作品，各小组课后完善作品并提交到教学平台。

五、知识巩固

1、总结知识点，使用教学平台中的随堂练习题巩固本所学知识。

2、使用教学平台中的测试题给学生布置作业。

拓展作业	1、学习微课《第四章内容介绍》； 2、课程平台拓展作业。
教学后记	