**第四章 探究碎片**

**1. 什么是碎片**

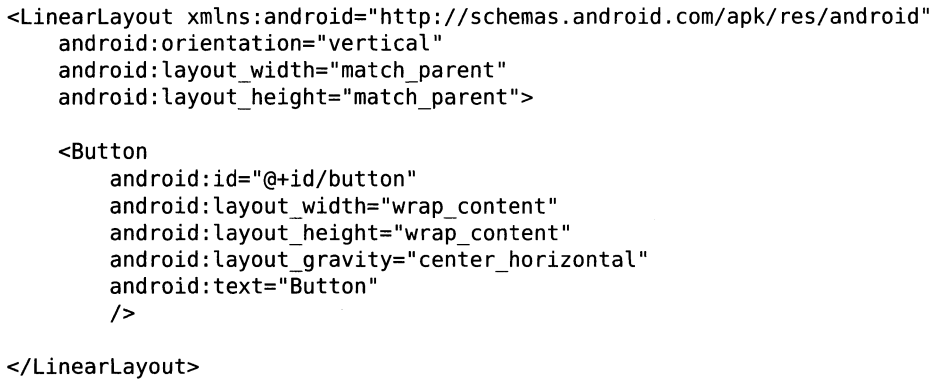
**碎片( Fragment)是一种可以嵌入在活动当中的 UI片段,它能让程序更加合理和充分地利用大屏幕的空间,因而在平板上应用得非常广泛。**

**2.碎片的使用方法**

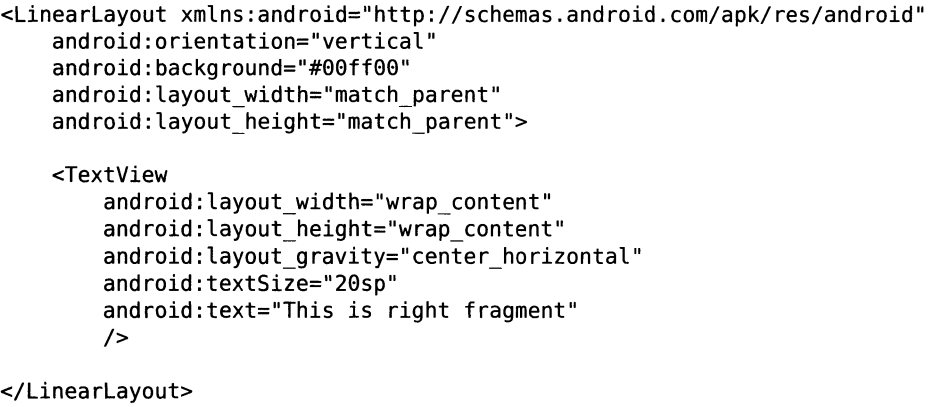
**先创建一个平板模拟器，接着新建一个FragmentTest项目**

**（1）碎片的简单用法**

**在一个活动中添加两个碎片，并让其平分活动空间。新建一个左侧碎片布局left\_fragment.xml，代码如下所示：**

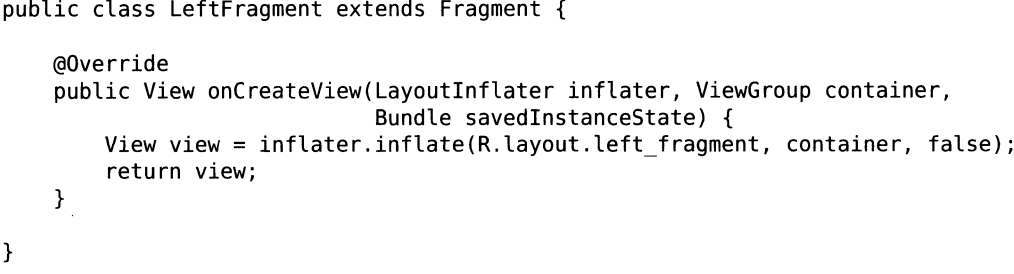


**再新建一个右侧布局碎片right\_fragment.xml,代码如下所示：**



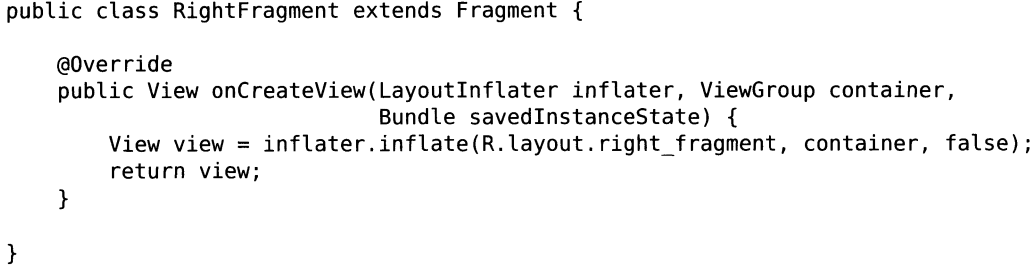
**把布局背景设置为绿色，便于和左侧区分。**

**接着新建一个leftFragment类让它继承自Fragment，代码如下所示：**



**这里仅仅是重写了 Fragment的 onCreateView()方法,然后在这个方法中通过Layoutlnflater的inflate( )方法将刚才定义的1eft\_fragment布局动态加载了进来。**

**再用同样的方法再新建一个Right Fragment,代码如下所示：**



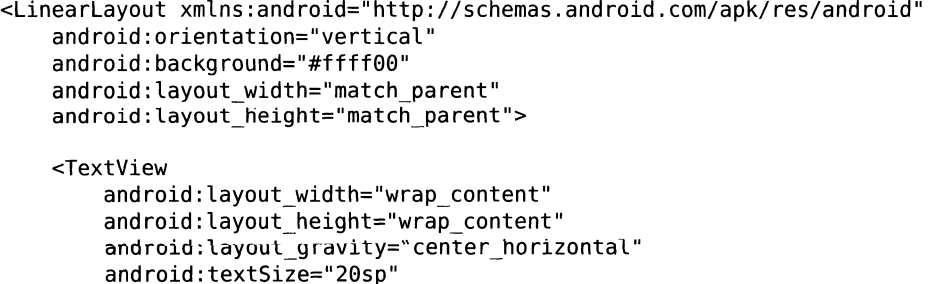
**接下来修改 activity\_main.xm1中的代码，如下所示：**

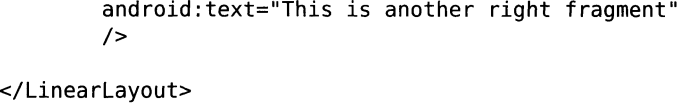


**使用<fragment>标签在布局中添加碎片,这里还需要通过 android:name属性来显式指明要添加的碎片类名,一定要将类的包名也加上。**

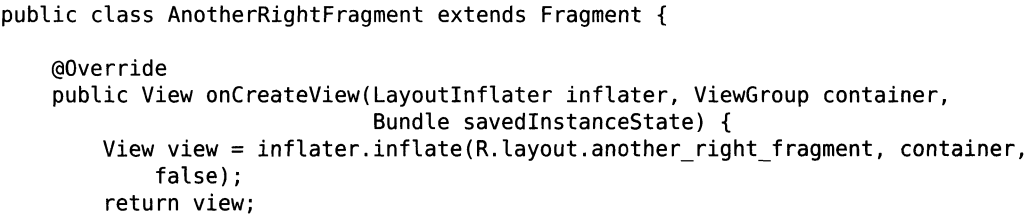
**（2）动态添加碎片**

**在上一节代码的基础上继续完善,新建 another\_right\_fragment.xm1,代码如下所示:**





**这里将背景色改为了黄色，新建 AnotherRightFragment作为另一个右侧碎片, 代码如下所示:**



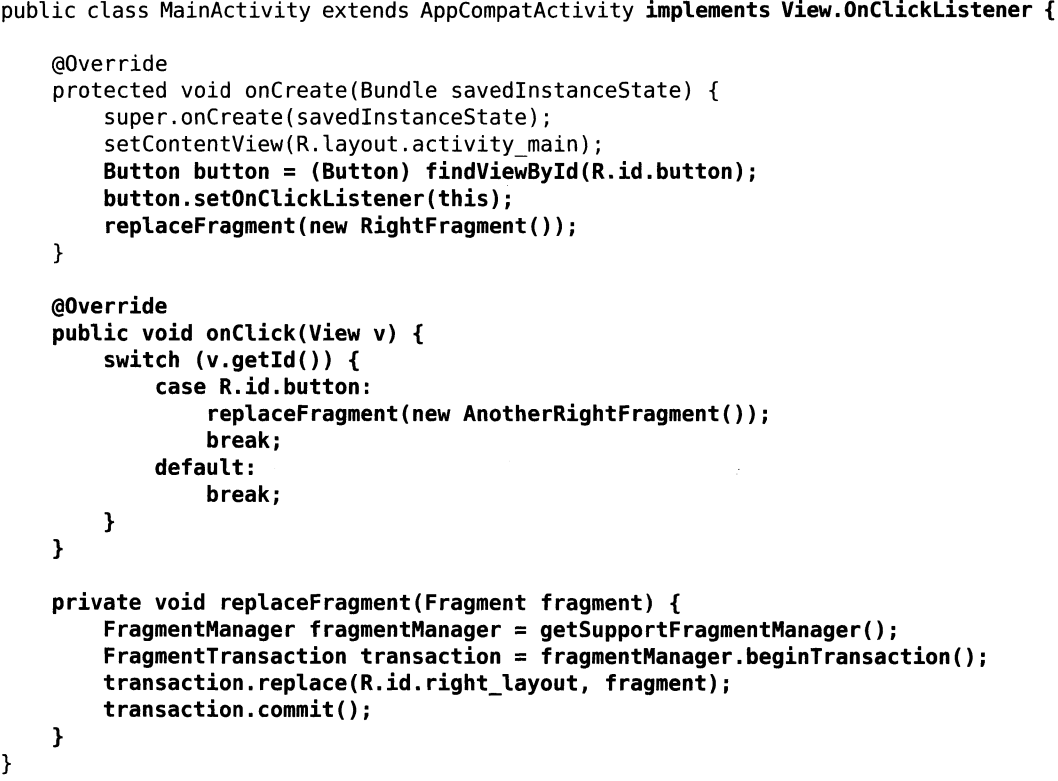
**}**

**}**

**这里在 onCreateView( )方法中加载刚刚创建的 another\_right\_ftagment布局。接下来将它动态的添加到活动中，修改Activity\_main.xml,代码如下所示：**



**在代码中向 FrameLayout 里添加内容,实现动态添加碎片的功能。修改MainActivity 中的代码, 如下所示:**



**这里首先给左侧碎片中的按钮注册了一个点击事件, 然后调用replaceFragment( )方法动态添加了RightFragment这个碎片。当点击左侧碎片中的按钮时, 又会调用replaceFragment()方法将右侧碎片替换成 AnotherRightFragment。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**动态添加碎片主要分为5步：**

**1)创建待添加的碎片实例。**

**2)获取 FragmentManager,在活动中可以直接通过调用 getsupportFragmentManager( )方法得到。**

**3)开启一个事务, 通过调用 beginTransaction( )方法开启。**

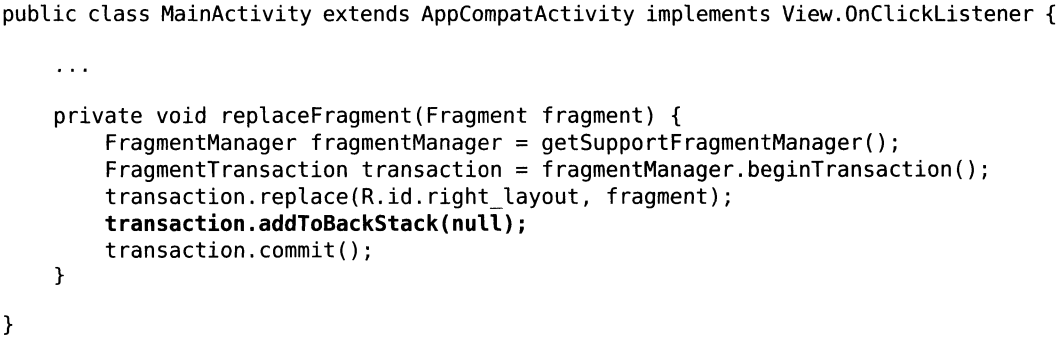
**4)向容器内添加或替換碎片, 一般使用 replace()方法实现,需要传入容器的 id和待添加的碎片实例。**

**5)提交事务,调用 commit()方法来完成。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**（3）在碎片中模拟返回栈**

**FragmentTransaction中提供了一个 addToBackStack( )方法, 可以用于将一个事务添加到返回桟中, 修改 MainActivity 中的代码, 如下所示:**



**这里在事务提交之前调用了 FragmentTransaction的 addToBackStack( )方法,它可以接收一个名字用于描述返回栈的状态, 一般传入null即可。**

**（4）碎片和活动之间进行通信**

**为了方便碎片和活动之间进行通信, FragmentManager提供了一个类似于 findViewById()的方法, 专门用于从布局文件中获取碎片的实例,代码如下所示:**



**调用 FragmentManager的 findFragmentById( )方法, 可以在活动中得到相应碎片的实例,** **然后就可以调用碎片里的方法了。**

**在碎片中也可以调用活动里的方法，是通过调用getActivity()方法来得到和当前碎片相关联的活动实例，代码如下所示：**



**3.碎片的生命周期**

**（1）碎片的状态和回调**

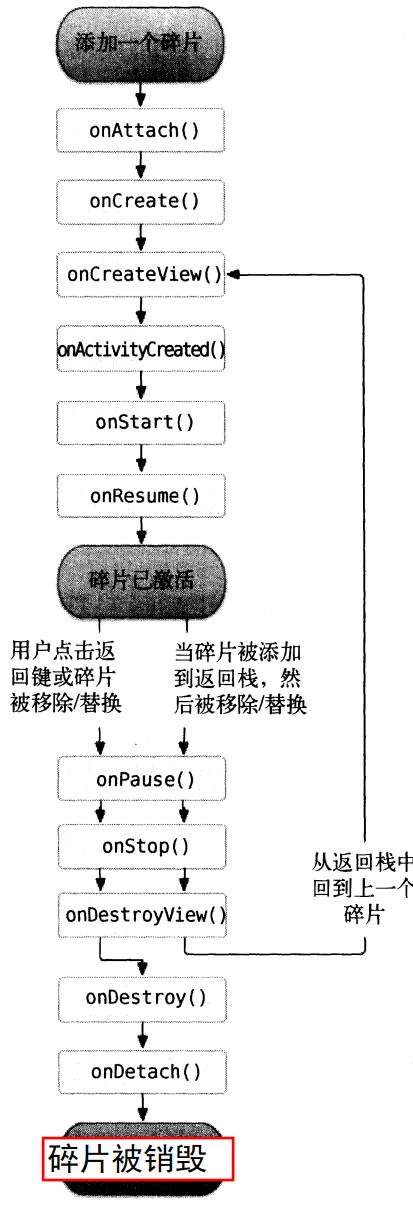
**碎片和活动类似也有4中运行状态**

1. **运行状态**
2. **暂停状态**
3. **停止状态**
4. **销毁状态 碎片总是依附于活动而存在的，因此当活动被销毁时，与它相关联的碎片就会进入销毁状态。**

**Fragment类中提供了一系列回调方法：**

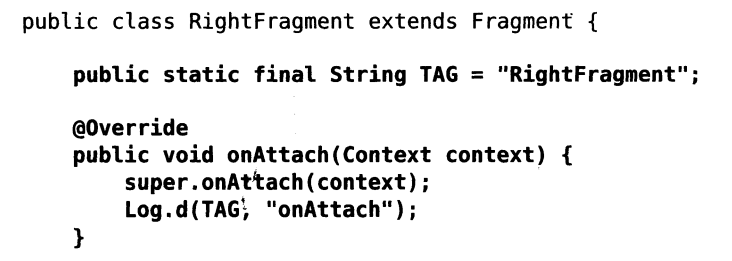
1. **onAttach()。当碎片和活动建立关联的时候调用。**
2. **oncreateview()。为碎片创建视图(加载布局)时调用。**
3. **onActivityCreated( ) 。确保与碎片相关联的活动一定已经创建完毕的时候调用。**
4. **onDestroyView()。当与碎片关联的视图被移除的时候调用。**
5. **onDetach()。当碎片和活动解除关联的时候调用。**

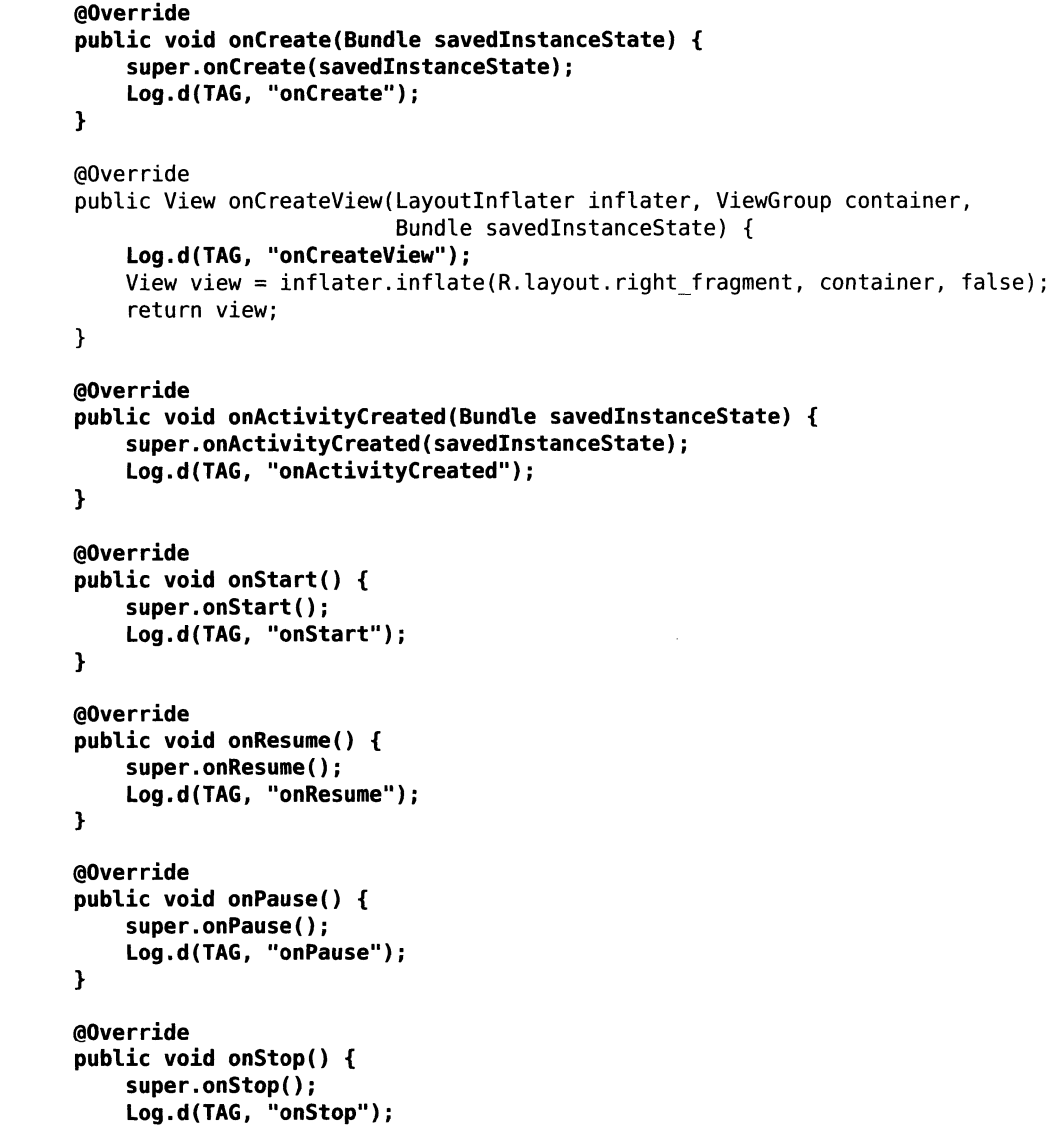
**碎片完整的生命周期示意图如下：**

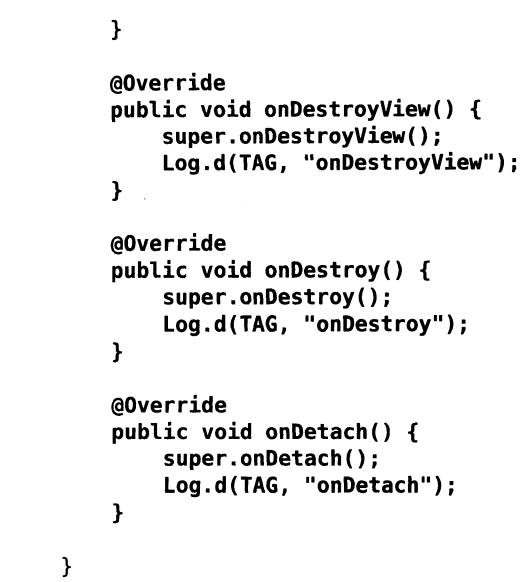


**（2）体验碎片的生命周期**

**通过例子来体验，在FragmentTest项目的基础上做改动，修改RightFragment中的代码如下所示：**







**当 R1ghtFragment第一次被加载到屏幕上时,会依次执行 onAttach()、onCreate( )、onCreateView( )、 onActivityCreated()、onStart( )和onResume()方法。然后点击 LeftFragment中的按钮，就会依次执行**

**onpause( )、onstop( )和onDestroyview( )方法。这是由于AnotherRightFragment替換了 RightFragment, 此时的 RightFragment进入了停止状态。**

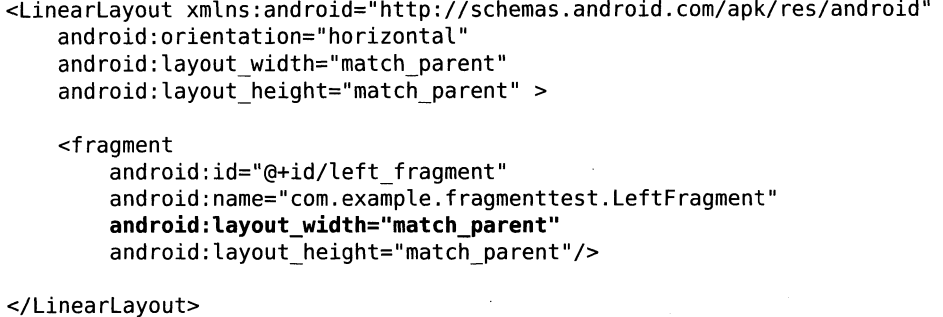
**4. 动态加载布局**

**动态加载布局只程序能够根据设备的分片率和屏幕大小在运行时来决定加载那个布局。**

**（1）使用限定符（Qualifiers）**

**借助限定符可以在运行时判断程序应该是使用双页模式还是単页模式。修改FragmentTest项目中的**

**activity\_main.xml文件，代码如下所示:**

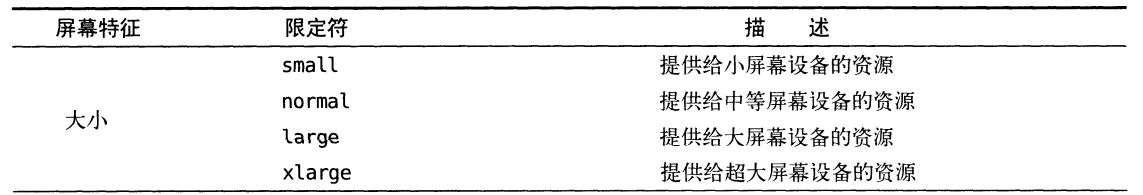


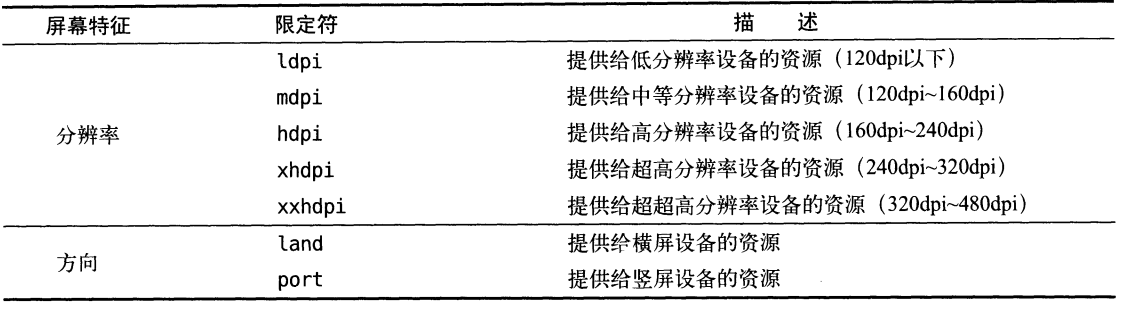
**接着在 res目录下新建1ayout-1arge文件夹,在这个文件夹下新建一个布局,也叫作 activity\_main.xml,代码如下所示：**



**layout/activity\_main布局只包含了一个碎片, 即单页模式, 而1ayout\_1arge/activity\_main布局包含了两个碎片,即双页模式。其中Large就是一个限定符,那些屏幕被认为是1La rge的设备就会自动加载1ayout-1arge文件夹下的布局,而小屏幕的设备则还是会加载layout文件夹下的布局。**

**Android中常见的一些限定符：**





**（2）使用最小宽度限定符（Smallest-width Qualifier）**

**最小宽度限定符允许我们对屏幕的宽度指定一个最小值,然后以这个最小值为临界点,屏幕宽度大于这个值的设备就加载一个布局, 屏幕宽度小于这个值的设备就加载另一个布局。在 res目录下新建layout-sw600dp文件夹,然后在这个文件夹下新建 activity\_main.xm1布局,** **代码如下所示:**



**当程序运行在屏幕宽度大于600dp的设备上时,会加载layout-sw600dp/activity\_main布局,当程序运行在屏幕宽度小于600dp的设备上时,则仍然加载默认的1ayout/activity\_main布局。**

**5.小结**

**\*了解了碎片的基本概念以及使用场景**

**\*掌握了碎片的常见用法**

**\*学习了碎片生命周期的相关内容以及动态加载布局的技巧**