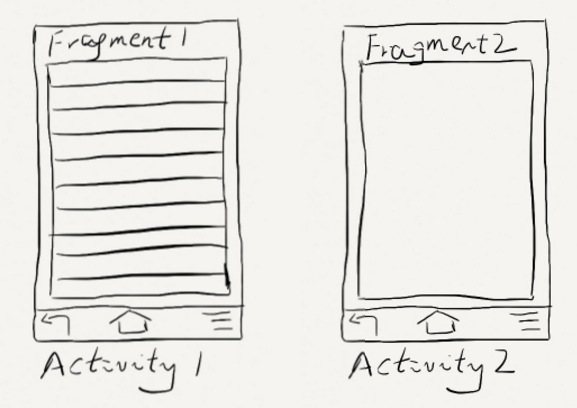
|  |
| --- |
| 1、Fragment是一个类似于activity的碎片，内部包含布局，有生命周期。  起初是为适配平板而设计的。  2、可以布局xml文件中使用fragment，也可以通过代码动态添加Fragment。  getFragmentManager().beginTransaction().replace(R.id.main\_layout, fragment1).commit();  3、生命周期  绑定-->初始化F-->创建F视图-->A执行onCreate回调  onStart() --> onResume() --> onPause() --> onStop()  销毁F视图 --> 销毁F --> 解绑。  onAttach()： 完成Fragment和Activity的绑定关联，参数中的Activity即为要绑定的Activity，可以进行赋值等操作；  onCreate() : 完成Fragment的初始化；  onCreateView() : 加载Fragment布局，绑定布局文件，创建视图；  onActivityCreated() : 表名与Fragment绑定的Activity已经执行完成了onCreate，可以与Activity进行交互操作。  onStart() : Fragment变为可见状态  onResume() : Fragment变为可交互状态  onPause()： Fragment变为不可交互状态(不代表不可见)  onStop(): Fragment变为不可见状态  onDestroyView() : 销毁Fragment的有关视图，但并未和Activity解绑，可以通过onCreateView()重新创建视图。  onDestroy() : 销毁Fragment时调用。  onDetach() : 解除和Activity的绑定。Fragmen销毁最后一步  onSaveInstanceState()：保存当前Fragment的状态。记录一些数据，比如EditText键入的文本，即使Fragment被回收又重新创建，一样能恢复EditText之前键入的文本。 |

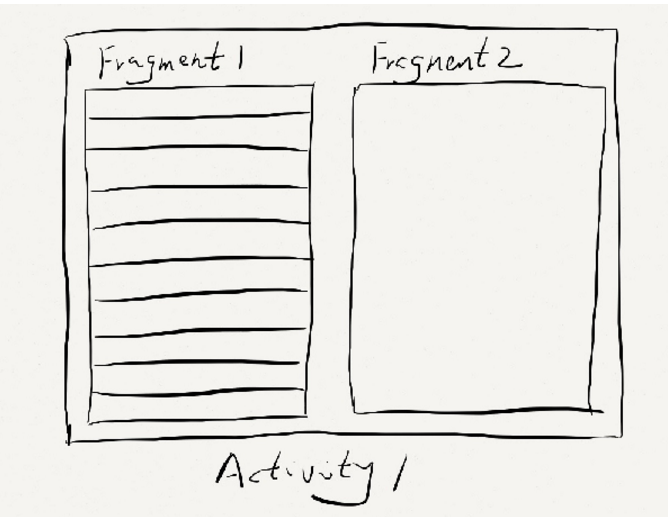
# 1、Fragment初探

为了让界面可以在平板上更好地展示，Android在3.0版本引入了Fragment(碎片)功能，它非常类似于Activity，可以像Activity一样包含布局。Fragment通常是嵌套在Activity中使用的，现在想象这种场景：有两个Fragment，Fragment 1包含了一个ListView，每行显示一本书的标题。Fragment 2包含了TextView和ImageView，来显示书的详细内容和图片。

如果现在程序运行竖屏模式的平板或手机上，Fragment 1可能嵌入在一个Activity中，而Fragment 2可能嵌入在另一个Activity中，如下图所示：



而如果现在程序运行在横屏模式的平板上，两个Fragment就可以嵌入在同一个Activity中了，如下图所示：



由此可以看出，使用Fragment可以让我们更加充分地利用平板的屏幕空间，下面我们一起来探究下如何使用Fragment。

首先需要注意，Fragment是在3.0版本引入的，如果你使用的是3.0之前的系统，需要先导入android-support-v4的jar包才能使用Fragment功能。

新建一个项目叫做Fragments，然后在layout文件夹下新建一个名为fragment1.xml的布局文件：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#00ff00" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is fragment 1"

android:textColor="#000000"

android:textSize="25sp" />

</LinearLayout>

可以看到，这个布局文件非常简单，只有一个LinearLayout，里面加入了一个TextView。我们如法炮制再新建一个fragment2.xml ：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#ffff00" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is fragment 2"

android:textColor="#000000"

android:textSize="25sp" />

</LinearLayout>

然后新建一个类Fragment1，这个类是继承自Fragment的：

public class Fragment1 extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment1, container, false);

}

}

我们可以看到，这个类也非常简单，主要就是加载了我们刚刚写好的fragment1.xml布局文件并返回。同样的方法，我们再写好Fragment2 ：

public class Fragment2 extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment2, container, false);

}

}

然后打开或新建activity\_main.xml作为主Activity的布局文件，在里面加入两个Fragment的引用，使用android:name前缀来引用具体的Fragment：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:baselineAligned="false" >

<fragment

android:id="@+id/fragment1"

android:name="com.example.fragmentdemo.Fragment1"

android:layout\_width="0dip"

android:layout\_height="match\_parent"

android:layout\_weight="1" />

<fragment

android:id="@+id/fragment2"

android:name="com.example.fragmentdemo.Fragment2"

android:layout\_width="0dip"

android:layout\_height="match\_parent"

android:layout\_weight="1" />

</LinearLayout>

最后打开或新建MainActivity作为程序的主Activity，里面的代码非常简单，都是自动生成的：

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

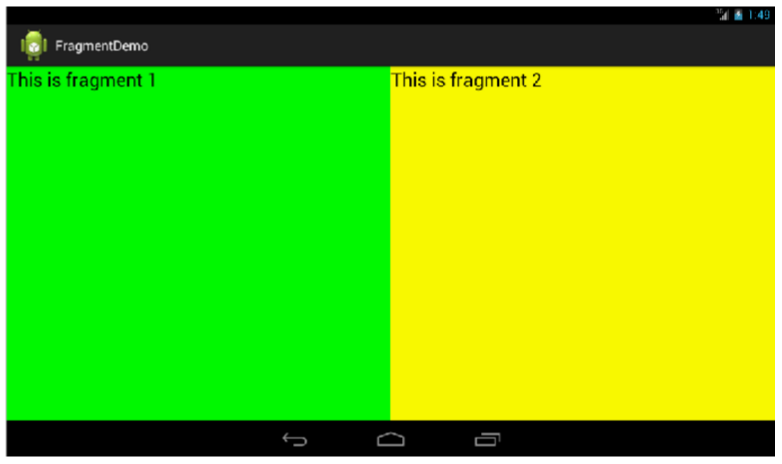
super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

现在我们来运行一次程序，就会看到，一个Activity很融洽地包含了两个Fragment，这两个Fragment平分了整个屏幕，效果图如下：



# 2、动态添加Fragment

你已经学会了如何在XML中使用Fragment，但是这仅仅是Fragment最简单的功能而已。Fragment真正的强大之处在于可以动态地添加到Activity当中，因此这也是你必须要掌握的东西。当你学会了在程序运行时向Activity添加Fragment，程序的界面就可以定制的更加多样化。下面我们立刻来看看，如何动态添加Fragment。

还是在上一节代码的基础上修改，打开activity\_main.xml，将其中对Fragment的引用都删除，只保留最外层的LinearLayout，并给它添加一个id，因为我们要动态添加Fragment，不用在XML里添加了，删除后代码如下：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/main\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:baselineAligned="false" >

</LinearLayout>

然后打开MainActivity，修改其中的代码如下所示：

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Display display = getWindowManager().getDefaultDisplay();

if (display.getWidth() > display.getHeight()) {

Fragment1 fragment1 = new Fragment1();

getFragmentManager().beginTransaction().replace(R.id.main\_layout, fragment1).commit();

} else {

Fragment2 fragment2 = new Fragment2();

getFragmentManager().beginTransaction().replace(R.id.main\_layout, fragment2).commit();

}

}

}

首先，我们要获取屏幕的宽度和高度，然后进行判断，如果屏幕宽度大于高度就添加fragment1，如果高度大于宽度就添加fragment2。动态添加Fragment主要分为4步：

1.获取到FragmentManager，在Activity中可以直接通过getFragmentManager得到。

2.开启一个事务，通过调用beginTransaction方法开启。

3.向容器内加入Fragment，一般使用replace方法实现，需要传入容器的id和Fragment的实例。

4.提交事务，调用commit方法提交。

# 3、Fragment的生命周期

和Activity一样，Fragment也有自己的生命周期，理解Fragment的生命周期非常重要，我们通过代码的方式来瞧一瞧Fragment的生命周期是什么样的：

public class Fragment1 extends Fragment {

public static final String TAG = "Fragment1";

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

Log.d(TAG, "onCreateView");

return inflater.inflate(R.layout.fragment1, container, false);

}

@Override

public void onAttach(Activity activity) {

super.onAttach(activity);

Log.d(TAG, "onAttach");

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Log.d(TAG, "onCreate");

}

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

Log.d(TAG, "onActivityCreated");

}

@Override

public void onStart() {

super.onStart();

Log.d(TAG, "onStart");

}

@Override

public void onResume() {

super.onResume();

Log.d(TAG, "onResume");

}

@Override

public void onPause() {

super.onPause();

Log.d(TAG, "onPause");

}

@Override

public void onStop() {

super.onStop();

Log.d(TAG, "onStop");

}

@Override

public void onDestroyView() {

super.onDestroyView();

Log.d(TAG, "onDestroyView");

}

@Override

public void onDestroy() {

super.onDestroy();

Log.d(TAG, "onDestroy");

}

@Override

public void onDetach() {

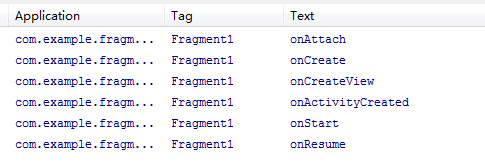
super.onDetach();

Log.d(TAG, "onDetach");

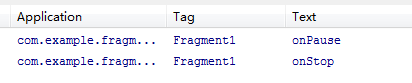
}

}

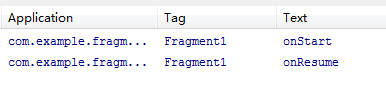
可以看到，上面的代码在每个生命周期的方法里都打印了日志，然后我们来运行一下程序，可以看到打印日志如下：



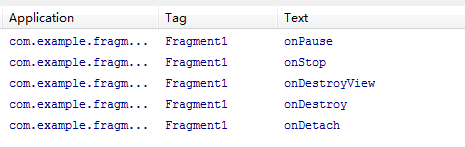
这时点击一下home键，打印日志如下：



如果你再重新进入进入程序，打印日志如下：



然后点击back键退出程序，打印日志如下：



看到这里，我相信大多数朋友已经非常明白了，因为这和Activity的生命周期太相似了。只是有几个Activity中没有的新方法，这里需要重点介绍一下：

onAttach方法：Fragment和Activity建立关联的时候调用。

onCreateView方法：为Fragment加载布局时调用。

onActivityCreated方法：当Activity中的onCreate方法执行完后调用。

onDestroyView方法：Fragment中的布局被移除时调用。

onDetach方法：Fragment和Activity解除关联的时候调用。

# 4、Fragment之间进行通信

通常情况下，Activity都会包含多个Fragment，这时多个Fragment之间如何进行通信就是个非常重要的问题了。我们通过一个例子来看一下，如何在一个Fragment中去访问另一个Fragment的视图。

还是在第一节代码的基础上修改，首先打开fragment2.xml，在这个布局里面添加一个按钮：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="#ffff00" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is fragment 2"

android:textColor="#000000"

android:textSize="25sp" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Get fragment1 text"

/>

</LinearLayout>

然后打开fragment1.xml，为TextView添加一个id：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#00ff00" >

<TextView

android:id="@+id/fragment1\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is fragment 1"

android:textColor="#000000"

android:textSize="25sp" />

</LinearLayout>

接着打开Fragment2.java，添加onActivityCreated方法，并处理按钮的点击事件：

public class Fragment2 extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment2, container, false);

}

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

Button button = (Button) getActivity().findViewById(R.id.button);

button.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

TextView textView = (TextView) getActivity().findViewById(R.id.fragment1\_text);

Toast.makeText(getActivity(), textView.getText(), Toast.LENGTH\_LONG).show();

}

});

}

}

现在运行一下程序，并点击一下fragment2上的按钮，效果如下图所示。

我们可以看到，在fragment2中成功获取到了fragment1中的视图，并弹出Toast。这是怎么实现的呢？主要都是通过getActivity这个方法实现的。getActivity方法可以让Fragment获取到关联的Activity，然后再调用Activity的findViewById方法，就可以获取到和这个Activity关联的其它Fragment的视图了。

# 5、除此之外，使用Fragment还有这么几个方面优势：

代码复用。特别适用于模块化的开发，因为一个Fragment可以被多个Activity嵌套，有个共同的业务模块就可以复用了，是模块化UI的良好组件。

Activity用来管理Fragment。Fragment的生命周期是寄托到Activity中，Fragment可以被Attach添加和Detach释放。

可控性。Fragment可以像普通对象那样自由的创建和控制，传递参数更加容易和方便，也不用处理系统相关的事情，显示方式、替换、不管是整体还是部分，都可以做到相应的更改。

Fragments是view controllers，它们包含可测试的，解耦的业务逻辑块，由于Fragments是构建在views之上的，而views很容易实现动画效果，因此Fragments在屏幕切换时具有更好的控制。

# 6、Fragment是什么

说了半天的Fragment，也看到这么多次Fragment这个名词出现，那么Fragment到底是什么东东呢？定义又是如何？

Fragment也可以叫为“片段”，但我觉得“片段”中文叫法有点生硬，还是保持叫Fragment比较好，它可以表示Activity中的行为或用户界面部分。我们可以在一个Activity中用多个Fragment组合来构建多窗格的UI，以及在多个Activity中重复使用某个Fragment。它有自己的生命周期，能接受自己的输入，并且可以在 Activity 运行时添加或删除Fragment（有点像在不同 Activity 中重复使用的“子 Activity”）。

简单来说，Fragment其实可以理解为一个具有自己生命周期的控件，只不过这个控件又有点特殊，它有自己的处理输入事件的能力，有自己的生命周期，又必须依赖于Activity，能互相通信和托管。