Android Fragment 完全解析，关于碎片你所需知道的一切

视频教程：http://www.jikexueyuan.com/study/index/cid/5/lid/5.html?hmsr=wenku_androidfragment

Swift 教程 安卓开发教程 ios 开发教程 cocos2dx 教程 手机应用开发 游戏开发教程

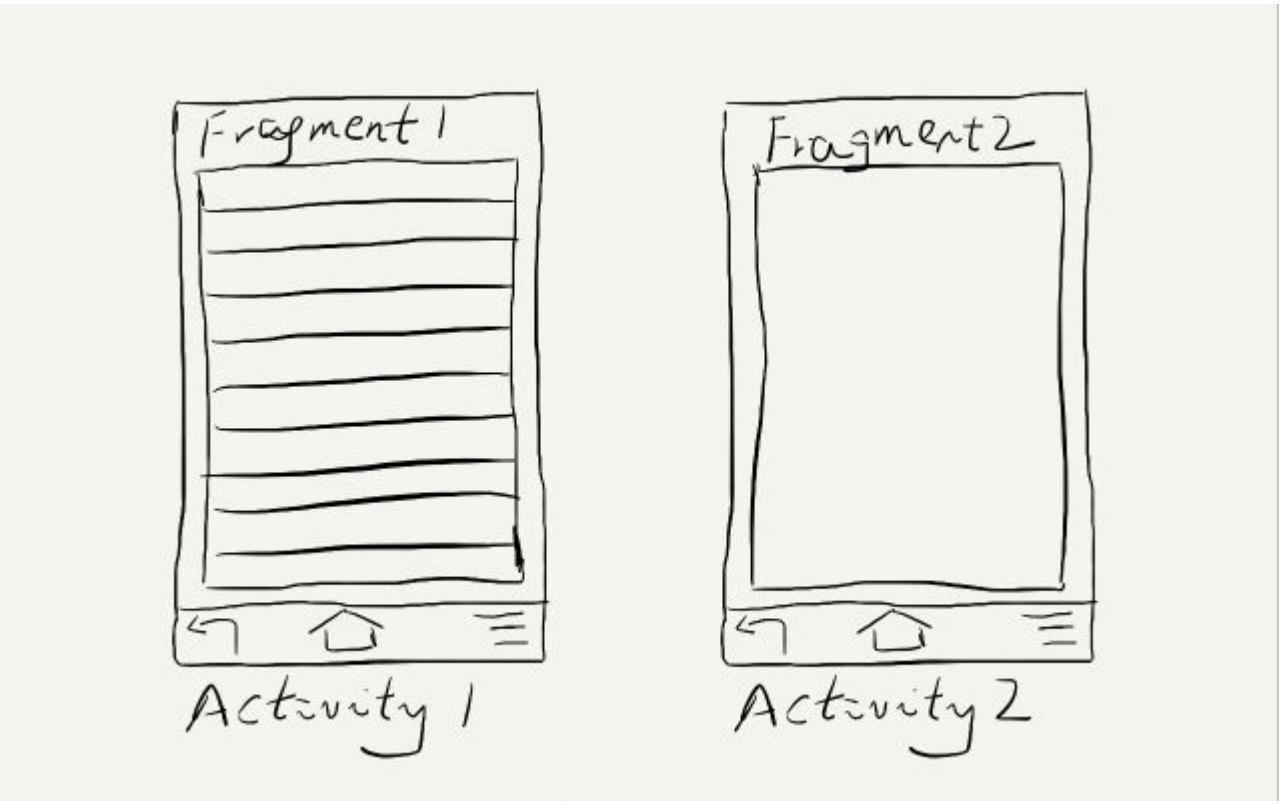
我们都知道，Android 上的界面展示都是通过 Activity 实现的，Activity 实在是太常用了，我相信大家都已经非常熟悉了，这里就不再赘述。

但是 Activity 也有它的局限性，同样的界面在手机上显示可能很好看，在平板上就未必了，因为平板的屏幕非常大，手机的界面放在平板上可能会有过分被拉长、控件间距过大等情况。这个时候更好的体验效果是在 Activity 中嵌入"小 Activity"，然后每个"小 Activity"又可以拥有自己的布局。因此，我们今天的主角 Fragment 登场了。

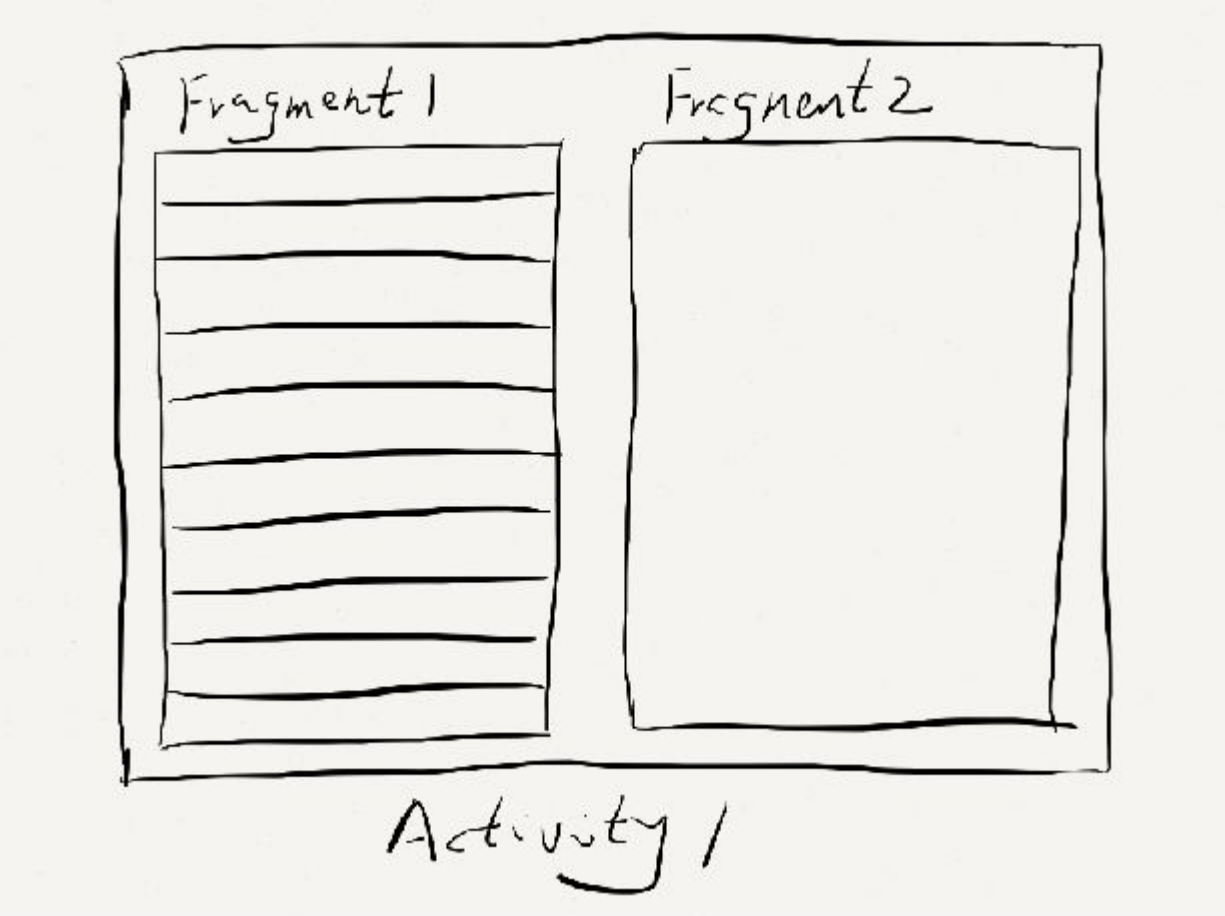
Fragment 初探

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：

[html]view plaincopy

```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2  android:layout_width="match_parent"
3  android:layout_height="match_parent"
4  android:background="#00ff00" >
5
6  <TextView
7  android:layout_width="wrap_content"
```

```
8   android:layout_height="wrap_content"
9   android:text="This is fragment 1"
10  android:textColor="#000000"
11  android:textSize="25sp" />
12
13  </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 ：

[html]view plaincopy

```
14  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
15  android:layout_width="match_parent"
16  android:layout_height="match_parent"
17  android:background="#ffff00" >
18
19  <TextView
20  android:layout_width="wrap_content"
21  android:layout_height="wrap_content"
22  android:text="This is fragment 2"
23  android:textColor="#000000"
24  android:textSize="25sp" />
25
26  </LinearLayout>
```

然后新建一个类 Fragment1，这个类是继承自 Fragment 的：

[java]view plaincopy

```
27  public class Fragment1 extends Fragment {
28
29  @Override
30  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
```

```
savedInstanceState) {
31  return inflater.inflate(R.layout.fragment1, container, false);
32  }
33
34  }
```

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 ：

[java]view plaincopy

```
35  public class Fragment2 extends Fragment {
36
37  @Override
38  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
39  return inflater.inflate(R.layout.fragment2, container, false);
40  }
41
42  }
```

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：

[html]view plaincopy

```
43  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
44  android:layout_width="match_parent"
45  android:layout_height="match_parent"
46  android:baselineAligned="false" >
47
48  <fragment
49  android:id="@+id/fragment1"
50  android:name="com.example.fragmentdemo.Fragment1"
51  android:layout_width="0dip"
```

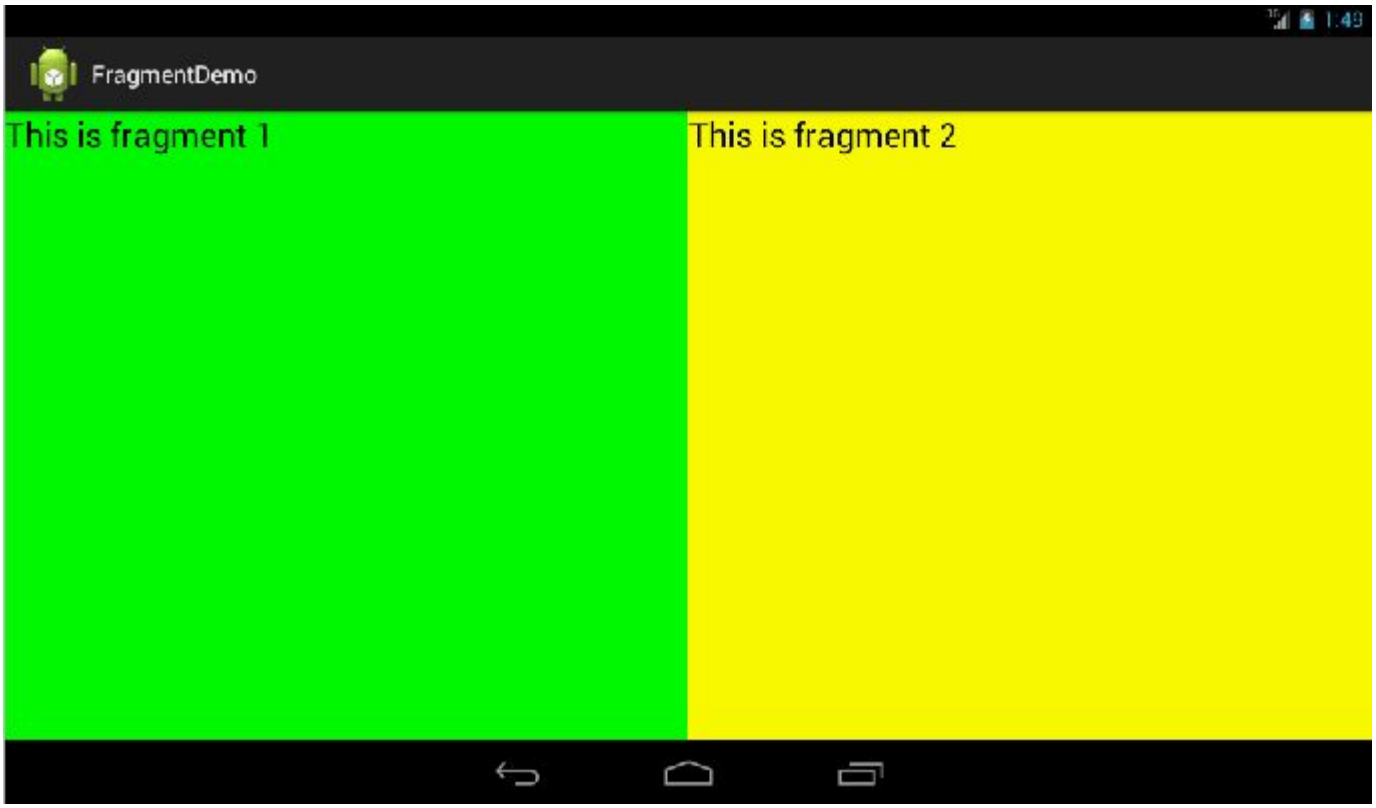
```
52  android:layout_height="match_parent"
53  android:layout_weight="1" />
54
55  <fragment
56  android:id="@+id/fragment2"
57  android:name="com.example.fragmentdemo.Fragment2"
58  android:layout_width="0dip"
59  android:layout_height="match_parent"
60  android:layout_weight="1" />
61
62  </LinearLayout>
```

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：

[java]view plaincopy

```
63  public class MainActivity extends Activity {
64
65  @Override
66  protected void onCreate(Bundle savedInstanceState) {
67  super.onCreate(savedInstanceState);
68  setContentView(R.layout.activity_main);
69  }
70
71  }
```

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：



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：

[html]view plaincopy

```
72  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
73  android:id="@+id/main_layout"
74  android:layout_width="match_parent"
75  android:layout_height="match_parent"
76  android:baselineAligned="false" >
77
78  </LinearLayout>
```

然后打开 MainActivity，修改其中的代码如下所示：

[java]view plaincopy

```
79  public class MainActivity extends Activity {
80
81  @Override
82  protected void onCreate(Bundle savedInstanceState) {
83  super.onCreate(savedInstanceState);
84  setContentView(R.layout.activity_main);
85  Display display = getWindowManager().getDefaultDisplay();
86  if (display.getWidth() > display.getHeight()) {
87  Fragment1 fragment1 = new Fragment1();
88  getFragmentManager().beginTransaction().replace(R.id.main_layout, fragment1).commit();
89  } else {
90  Fragment2 fragment2 = new Fragment2();
91  getFragmentManager().beginTransaction().replace(R.id.main_layout, fragment2).commit();
92  }
93  }
94
95  }
```

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：

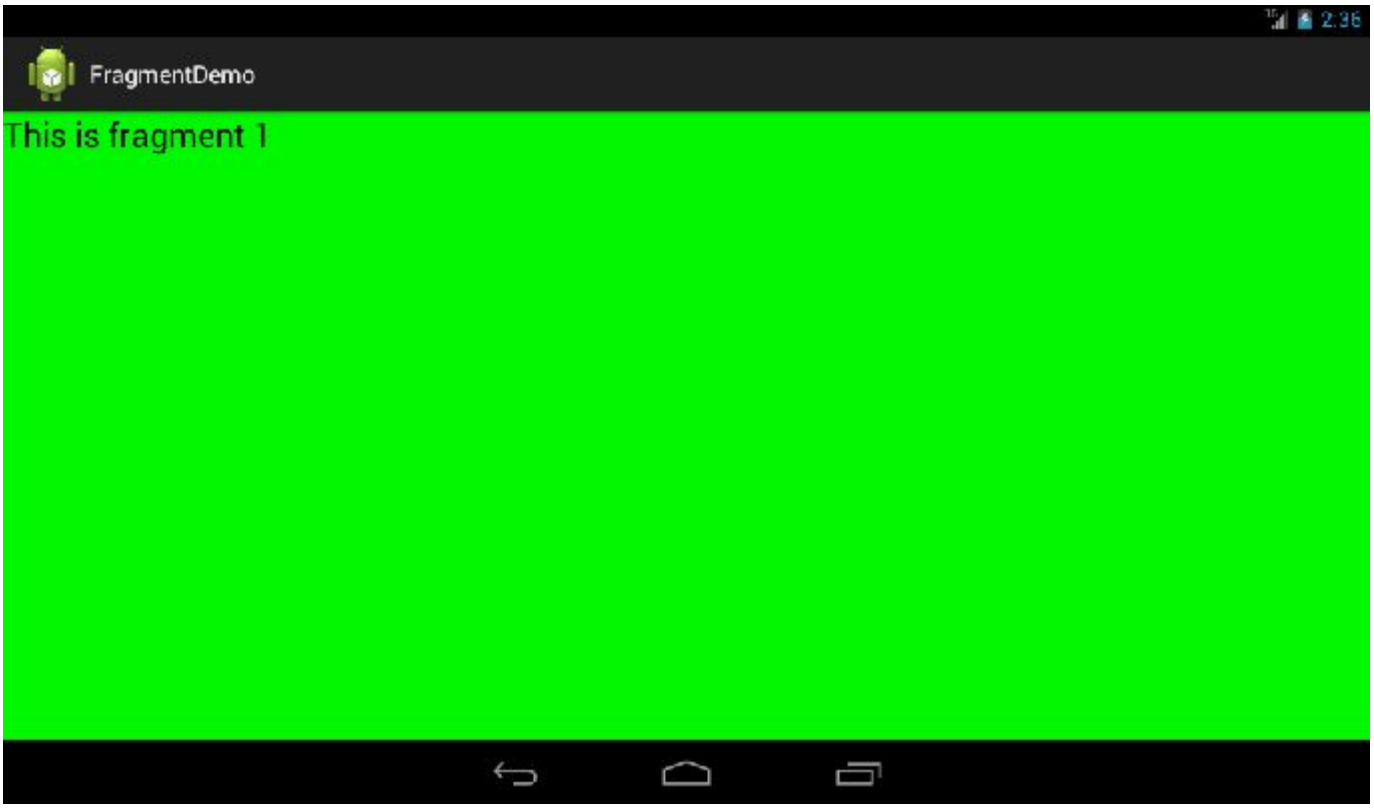
1.获取到 FragmentManager，在 Activity 中可以直接通过 getFragmentManager 得到。

2.开启一个事务，通过调用 beginTransaction 方法开启。

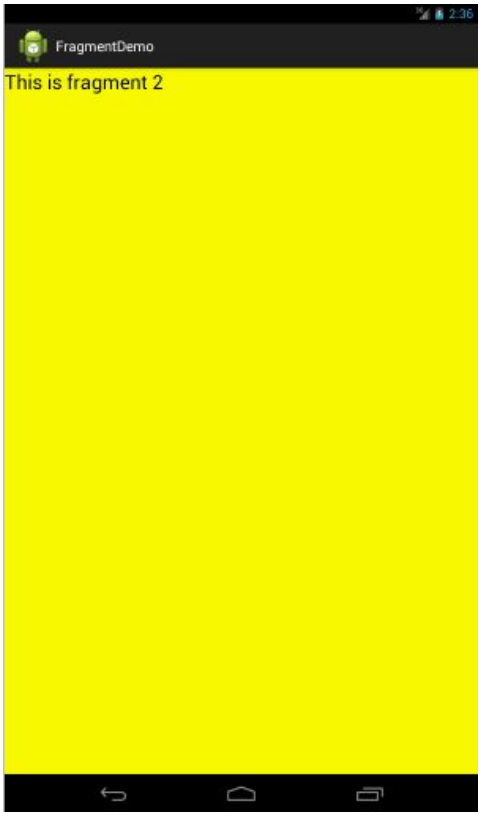
3.向容器内加入 Fragment，一般使用 replace 方法实现，需要传入容器的 id 和 Fragment 的实例。

4.提交事务，调用 commit 方法提交。

现在运行一下程序，效果如下图所示：



如果你是在使用模拟器运行，按下 ctrl + F11切换到竖屏模式。效果如下图所示：



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：

[java]view plaincopy

```
96  public class Fragment1 extends Fragment {
97  public static final String TAG = "Fragment1";
98
99  @Override
100 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
101 Log.d(TAG, "onCreateView");
102 return inflater.inflate(R.layout.fragment1, container, false);
103 }
104
105 @Override
106 public void onAttach(Activity activity) {
107 super.onAttach(activity);
108 Log.d(TAG, "onAttach");
109 }
110
111 @Override
112 public void onCreate(Bundle savedInstanceState) {
113 super.onCreate(savedInstanceState);
114 Log.d(TAG, "onCreate");
115 }
116
117 @Override
118 public void onActivityCreated(Bundle savedInstanceState) {
```

```
119 super.onActivityCreated(savedInstanceState);
120 Log.d(TAG, "onActivityCreated");
121 }
122
123 @Override
124 public void onStart() {
125 super.onStart();
126 Log.d(TAG, "onStart");
127 }
128
129 @Override
130 public void onResume() {
131 super.onResume();
132 Log.d(TAG, "onResume");
133 }
134
135 @Override
136 public void onPause() {
137 super.onPause();
138 Log.d(TAG, "onPause");
139 }
140
141 @Override
142 public void onStop() {
143 super.onStop();
144 Log.d(TAG, "onStop");
145 }
146
147 @Override
148 public void onDestroyView() {
```

```
149 super.onDestroyView();
150 Log.d(TAG, "onDestroyView");
151 }
152
153 @Override
154 public void onDestroy() {
155 super.onDestroy();
156 Log.d(TAG, "onDestroy");
157 }
158
159 @Override
160 public void onDetach() {
161 super.onDetach();
162 Log.d(TAG, "onDetach");
163 }
164
165 }
```

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：

Application	Tag	Text
com.example.fragm...	Fragment1	onAttach
com.example.fragm...	Fragment1	onCreate
com.example.fragm...	Fragment1	onCreateView
com.example.fragm...	Fragment1	onActivityCreated
com.example.fragm...	Fragment1	onStart
com.example.fragm...	Fragment1	onResume

这时点击一下 home 键，打印日志如下：

Application	Tag	Text
com.example.fragm...	Fragment1	onPause
com.example.fragm...	Fragment1	onStop

如果你再重新进入进入程序，打印日志如下：

Application	Tag	Text
com.example.fragm...	Fragment1	onStart
com.example.fragm...	Fragment1	onResume

然后点击 back 键退出程序，打印日志如下：

Application	Tag	Text
com.example.fragm...	Fragment1	onPause
com.example.fragm...	Fragment1	onStop
com.example.fragm...	Fragment1	onDestroyView
com.example.fragm...	Fragment1	onDestroy
com.example.fragm...	Fragment1	onDetach

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。

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：

[html]view plaincopy

```
166 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
167 android:layout_width="match_parent"
168 android:layout_height="match_parent"
169 android:orientation="vertical"
170 android:background="#ffff00" >
```

```
171
172 <TextView
173 android:layout_width="wrap_content"
174 android:layout_height="wrap_content"
175 android:text="This is fragment 2"
176 android:textColor="#000000"
177 android:textSize="25sp" />
178
179 <Button
180 android:id="@+id/button"
181 android:layout_width="wrap_content"
182 android:layout_height="wrap_content"
183 android:text="Get fragment1 text"
184 />
185
186 </LinearLayout>
```

然后打开 fragment1.xml，为 TextView 添加一个 id：

[html]view plaincopy

```
187 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
188 android:layout_width="match_parent"
189 android:layout_height="match_parent"
190 android:background="#00ff00" >
191
192 <TextView
193 android:id="@+id/fragment1_text"
194 android:layout_width="wrap_content"
195 android:layout_height="wrap_content"
196 android:text="This is fragment 1"
197 android:textColor="#000000"
```

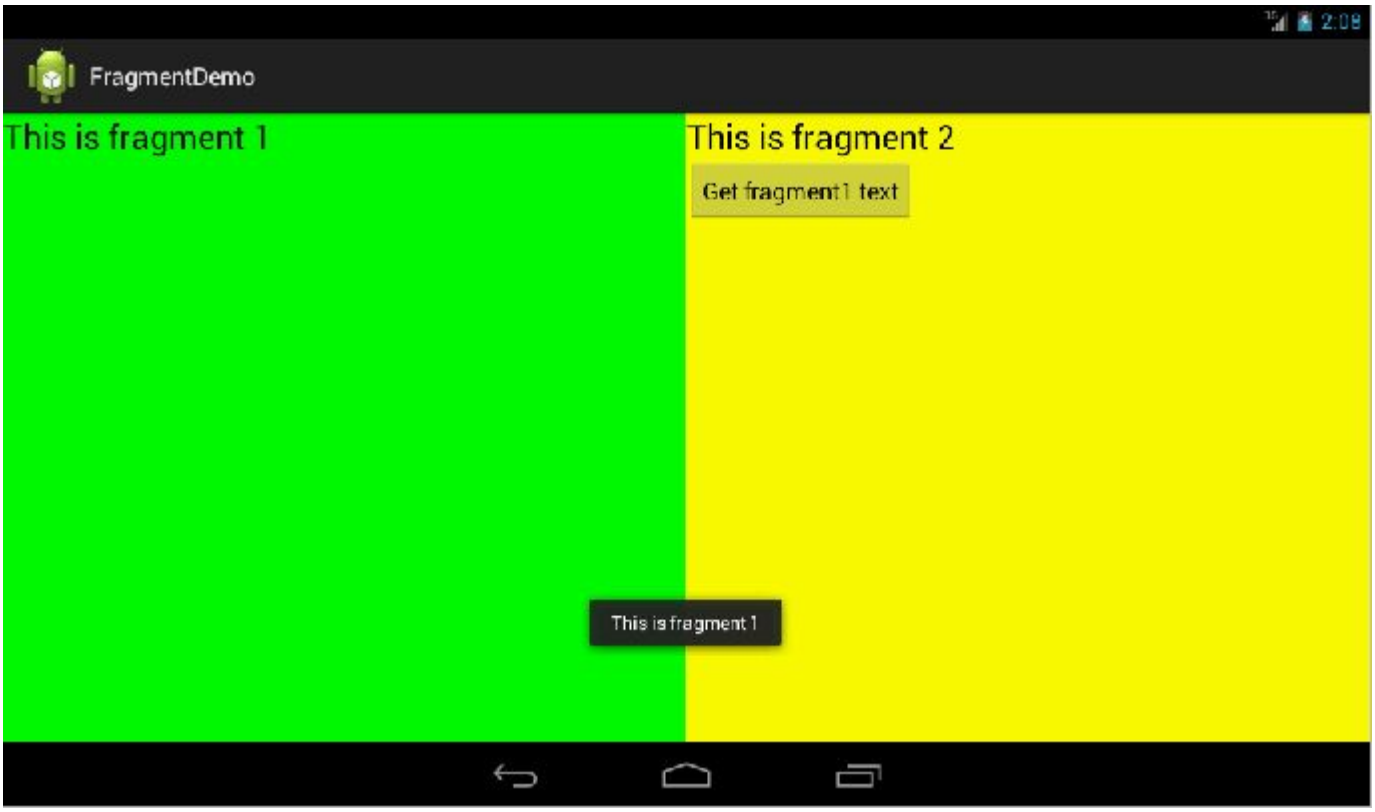
```
198 android:textSize="25sp" />
199
200 </LinearLayout>
```

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

[java]view plaincopy

```
201 public class Fragment2 extends Fragment {
202
203 @Override
204 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
205 return inflater.inflate(R.layout.fragment2, container, false);
206 }
207
208 @Override
209 public void onActivityCreated(Bundle savedInstanceState) {
210 super.onActivityCreated(savedInstanceState);
211 Button button = (Button) getActivity().findViewById(R.id.button);
212 button.setOnClickListener(new OnClickListener() {
213 @Override
214 public void onClick(View v) {
215 TextView textView = (TextView) getActivity().findViewById(R.id.fragment1_text);
216 Toast.makeText(getActivity(), textView.getText(), Toast.LENGTH_LONG).show();
217 }
218 });
219 }
220
221 }
```

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。

转载出处：http://blog.csdn.net/guolin_blog/article/details/8881711