**fragment的全讲解**

作者：牛奶晓蔬更新于 07月15日访问（384）评论（[7](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "comment)）

在一个Activity中，一个Fragment代表一种行为或者用户界面的一部分。你可以联合起来多个Fragment在一个Activity中创建多面板的UI，并且可以重用一个Fragment在多个activity中。你可以认为一个Fragment是一个Activity的一个模块部分，Fragment有它自己的生命周期，响应他自己的输入事件，并且你可以在Activity运行期间增加或者删除一个Fragment。

Fragment必须总是被嵌入到一个Activity中，并且它的生命周期直接受宿主Activity生命周期的影响。

Figure 1. Anexample of how two UI modules 2that are typically separated into two activitiescan be combined into one activity, using fragments.

Android 引进Fragment在 Android3.0（API Level "Honeycomb"）中，主要用来支持动态的，灵活的UI设计，让它运行在像平板电脑等大屏幕设备上。

在你的应用程序中， 一个Fragment应该是一个模块化、可重用的组件。那是因为Fragment能定义其自己的布局和自己的行为利用自身的生命周期回调方法,你可以在多个Activity中包括同一个Fragment。这是特别重要的,因为它允许你去适配你的用户去体验到的不同的屏幕尺寸。例如，如果屏幕尺寸足够大的话，可以放进去多个Fragment，如果不够大，可以用其他的activity去放其他的Fragment。

For example—to continue with the news applicationexample—the application can embed two fragments in Activity A, when running onan extra large screen (a tablet, for example). However, on a normal-sizedscreen (a phone, for example), there's not be enough room for both fragments,so Activity A includes only the fragment for the list of articles, and whenthe user selects an article, it starts Activity B, which includes the fragment toread the article. Thus, the application supports both design patterns suggestedin figure 1.

创建一个Fragment

去创建一个Fragment，要创建一个类，继承Fragment或者Fragment子类。Fragment的代码和Activity的代码非常相似。Fragment包括的回调方法和Activity相似的有 onCreate(), onStart(),onPause(), 和 onStop(). 事实上，如果你想把现有的程序改成使用Fragment，可能只需要简单把Activity中回调方法中的代码转移到Fragment相应的回调方法中去。

通常,你应该实现至少实现以下的生命周期方法:

onCreate()

当创建Fragment的时候，系统调用这个方法。在你的方法实现中，你要初始化你必须的组件。

onCreateView()

Fragment需要第一次绘制界面时，系统调用此方法。为了创建一个UI，必须返回一个View，这个View是从Fragment的Layout文件中得到的。如果你的Fragment不想提供UI的，就返回null

onPause()

当系统调用此方法时，第一表明用户已经离开了此Fragment。通常要在这个方法内提交发生的变化。  
大多数应用程序至少应实现这三种方法。

也有一些子类你需要去继承，而不是最基本的Fragment类：

DialogFragment

展现一个会飘动的Dialog。

ListFragment

展现一个条目列表。

PreferenceFragment

展现一个带有层级的配置列表。类似于PreferenceActivity 。为你的应用程序创建一个设置软件配置的类非常有帮助。

向fragment增加一个用户界面：

一个Fragment通常作为一个Activity的一部分，并把自己的布局贡献给这个Activity。

要想提供一个layout布局文件给Fragment，你必须实现 onCreateView() 回调方法。这个在Fragment要绘制界面时，系统调用。

注意：如果你的Fragment是ListFragment的子类，在onCreateView()中默认实现返回一个ListView,你并不需要实现onCreateView()方法。

为了返回一个布局在onCreateView()，你可以一个布局文件中返回，onCreateView()提供一个LayoutInflater 对象。

例子如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **public** **static** **class** **ExampleFragment** extendsFragment {  @Override  **public** View onCreateView(LayoutInflater inflater, ViewGroupcontainer,  Bundle savedInstanceState) {  *// Inflate the layout for thisfragment*  **return** inflater.inflate(R.layout.example\_fragment, container, **false**);  }  } |

container 参数是父 ViewGroup （从activity的layout文件中获得），你的Fragment将插入到这个 ViewGroup 中。

savedInstanceState 参数是一个 Bundle对象，提供之前的fragment实例的数据，当恢复fragment的时候。

inflate() 方法的三个参数：

The resource ID of the layout you want to inflate.

The ViewGroup to be the parent of the inflated layout. Passing the container is important in order for the system to apply layout parameters to the root view of the inflated layout, specified by the parent view in which it's going.

A boolean indicating whether the inflated layout should be attached to the ViewGroup (the second parameter) during inflation. (In this case, this is false because the system is already inserting the inflated layout into the container—passing true would create a redundant view group in the final layout.)

向一个activity增加fragment

通常fragment作为宿主activity界面的一部分，它嵌入到整个视图层次结构中，并作为其中的一部分。有两方式把fragment嵌入到activity布局当中去。

一、在activity布局文件中声明fragment

在这种情形中，可以把fragment当做view一样去是定属性，下面的例子是在一个activity布局文件中，嵌入了两个fragment

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?xml version="1.0" encoding="utf-8"?>  **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"**>**  **<fragment** android:name="com.example.news.ArticleListFragment"  android:id="@+id/list"  android:layout\_weight="1"  android:layout\_width="0dp"  android:layout\_height="match\_parent" **/>**  **<fragment** android:name="com.example.news.ArticleReaderFragment"  android:id="@+id/viewer"  android:layout\_weight="2"  android:layout\_width="0dp"  android:layout\_height="match\_parent" **/>**  **</LinearLayout>** |

当系统创建了activity布局文件之后，并实例化定义的每个fragment，调用每个fragment的 onCreateView() 方法，恢复每个fragment的布局文件，系统通过fragment返回的View直接替代fragment元素。

注意：每个fragment需要一个唯一的标识，当activity重新启动时，系统需要使用这个唯一的标识去恢复这个fragment。下面有三种方式为fragment提供标识：

（1）提供android:id属性，设置id  
（2）提供android:tag 属性，设置唯一的字符串  
（3）如果前两种没有设置，系统使用 containerview 的id

二、编码方式向一个存在的 ViewGroup 增加fragment

在activity运行期间的任何时候，你都可以把fragment加入到activity布局当中去。你只需要简单的定义一个放置fragment的 ViewGroup 。为了确保在你的activity中处理fragment，像增加、删除、替换fragment，你必须使用FragmentTransaction 的API，在activity中你可以这样获得FragmentTransaction 实例：

FragmentManager fragmentManager = getFragmentManager()  
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

在这时，你可以使用add()方法加入进去一个fragment，指明把fragment加入到那个View中去，例如：

ExampleFragment fragment =new ExampleFragment();  
fragmentTransaction.add(R.id.fragment\_container,fragment);  
fragmentTransaction.commit();

Add()方法传进去的第一个参数是fragment需要放置到的那个 ViewGroup的资源id 。第二个参数是你要加进去的fragment

一旦你使用 FragmentTransaction 改变了东西，一定要调用commit() 方法才能起作用。

增加一个没有UI的fragment

上面都是增加带有UI的fragment，然而，你还可以使用没有UI界面，只具有后台行为的fragment。  
在activity中增加没有UI的fragment通过使用add(Fragment,String) 方法，提供一个唯一标识的字符串Tag，而不是view id，这种fragment不涉及activity中的布局，他不需要接收调用 onCreateView() 返回的值，因此你不需要实现此方法。

提供String Tag 不是必须的，对于没有UI的fragment来说。你也可以在带有ＵＩ的fragment中使用String Tag，然而String Tag是没有UI的fragment的唯一标识，如果你想在activity中得到fragment，你需要通过调用 findFragmentByTag() 方法得到。

关于使用没有UI的fragment做后台任务的activity，看这个例子： FragmentRetainInstance.java

管理Fragments

要在你的activity中管理fragment ，你需要使用 FragmentManager ， FragmentManager 在activity中通过调用getFragmentManager() 方法获得。

通过FragmentManager你可以做一些事情：

1）取出一个fragment，通过 findFragmentById() ， findFragmentByTag()  
2）使用 popBackStack() 方法，从后台堆栈中弹出一个fragment  
3）为后台堆栈注册能监听改变的监器， addOnBackStackChangedListener().

前面的讲解表明，你也可以通过 FragmentManager 去打开一个 FragmentTransaction ，  
FragmentTransaction 允许你执行事务，比如增加或者删除fragment。

执行Fragment的事务操作

在activity中使用fragment的最大特征是，fragment可以被增加，删除，替换或者执行其他的操作，在相应用户的交互的时候。

fragment的每一次改变的提交给activity，都称为一个事务，你可以使用FragmentTransaction 的API去操作事务。你可以把事务保存在activity管理的后台栈当中，允许用户返回浏览改变的fragment。

你可以像这样一样从FragmentManager 中获得一个 FragmentTransaction 实例：

FragmentManager fragmentManager = getFragmentManager();  
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

每一个事务是一系列你同时需要执行的改变。你可以建立你所有要求执行的事务操作，使用add(), remove(),and replace() 方法，最后一定要调用 commit(). 方法，提交你的事务。

在你调用 commit() 之前，你为了提交这个事务到fragment返回事务栈当中去，你可能会调用addToBackStack() ，这个返回栈由activity管理，允许用户按返回键回到之前的fragment状态。  
例子：这里是如果用一个fragment去替换另一个fragment，并把之前的状态保存到返回栈当中。

// Create new fragment andtransaction  
Fragment newFragment = new ExampleFragment();  
FragmentTransaction transaction = getFragmentManager().beginTransaction();

// Replace whatever is in thefragment\_container view with this fragment,  
// and add the transaction tothe back stack  
transaction.replace(R.id.fragment\_container,newFragment);  
transaction.addToBackStack(null);

// Commit the transaction  
transaction.commit();

在这个例子中newFragment替换了所有的在被R.id.fragment\_container标识的容器的fragment。通过调用addToBackStack() ，把替换事务提交到了返回事务栈中，此时用户可以回滚事务，按返回键返回到之前的fragment的状态。

如果你的多个改变放在同一个事务中，并且调用了 addToBackStack() ，在你调用commit() 之前，此时所有的改变都会作为一个事务增加到返回栈当中去，当按返回键之后，所有的改变都会回滚到原始状态。

增加所有的修改到 FragmentTransaction 的顺序是无关紧要的，除了最后你必须调用 commit() 方法和你增加多个fragment到容器当中，此时你增加的顺序会表现在视图的层次结构当中。

当执行一个事务，移除一个fragment时， 如果你没有调用 addToBackStack() ，那么当你提交这个事务的时候这个fragment将会被销毁，用户将会不能再返回去访问它。然而，如果你移除一个fragment，并且调用了 addToBackStack() ，此时fragment将处于stop状态，如果用户返回访问它时，这个fragment将会恢复。  
Tip ：你可以为你的fragment事务提交事务之前调用 setTransition() 设置一个转换动画，

调用commit() 方法不会立即执行事务，它运行在activity的UI线程的时间表中，直到UI线程运行到他。如果必要的话，你可以在UI线程中调用 executePendingTransactions() ，把commit() 提交的事务立即执行。通常情况下，做这个操作是没有必要的，除非这个事务依赖其他线程去执行业务。

警告：你只能在activity保存状态（当用户离开这个activity时）之前调用commit（）方法执行事务。如果在保存状态这个点之后提交事务，将会抛出异常，这是因为你的activity想要恢复的时候，提交的事务已经丢失。为了解决这种情况的问题，可以调用commitAllowingStateLoss() 这个方法。

与Activity通信

提别说明：一个fragment通过 getActivity() 可以访问宿主activity的实例，并且可以非常方便的获得activity布局文件中的视图组件等工作。

View listView = getActivity().findViewById(R.id.list);  
同样，宿主activity也可以通过FragmentManager的findFragmentById() 活 findFragmentByTag()方法获得fragment的引用：例如：

ExampleFragment fragment =(ExampleFragment) getFragmentManager().findFragmentById(R.id.example\_fragment);

为activity创建一个回调事件

在某些情况下,你需要fragment和activity共享事件,一个好的方式是:在一个fragment中定义一个回调接口,然后activity实现此接口.当这个activity接收到这个接口的回调时,如果需要的话,你可以共享你的数据与其他的fragment.

例如:有一个新闻应用程序,在一个activity中有两个fragment,一个用来显示新闻列表(fragment A),一个用来展示新闻内容（fragment B）,当一个新闻被选择的时候，fragment A要告诉activity，让fragmentB去展示新闻内容。在这种情况下， OnArticleSelectedListener 接口要在fragmentA中声明。

public static class FragmentA extendsListFragment {  
...  
// Container Activity must implementthis interface  
public interface OnArticleSelectedListener {  
public void onArticleSelected(UriarticleUri);  
}  
...  
}

此时，activity持有 OnArticleSelectedListener接口 的实现，并重写了onArticleSelected() 方法，把来自fragmentA的事件通知给fragmentB。为了确保宿主activity实现此接口，在fragmentA的回调方法 onAttach() 中，把同过 onAttach()方法传递过来的activity强制转型为一个OnArticleSelectedListener 实例。

public static class FragmentA extendsListFragment {  
OnArticleSelectedListener mListener;  
...  
@Override  
public void onAttach(Activity activity) {  
super.onAttach(activity);  
try {  
mListener = (OnArticleSelectedListener) activity;  
} catch (ClassCastException e) {  
throw new ClassCastException(activity.toString() + "must implement OnArticleSelectedListener");  
}  
}  
...  
}

如果这个activity没有实现这个接口，此时将抛出ClassCastException. 。成功的情况下，mListener 持有 实现OnArticleSelectedListener接口的activity，因此fragmentA可以通过Activity调用实现了OnArticleSelectedListener 接口的方法来共享事件。

举例说明：如果fragmentA继承了 ListFragment ，每当用户点击了一个条目，系统就会调用fragment中的 onListItemClick() 方法，fragment中的代码就会调用activity实现的onArticleSelected() 方法，fragment就能与activity共享事件了。

public static class FragmentA extendsListFragment {  
OnArticleSelectedListener mListener;  
...  
@Override  
public void onListItemClick(ListViewl, View v, int position, long id) {  
// Append the clicked item's row IDwith the content provider Uri  
Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT\_URI, id);  
// Send the event and Uri to the hostactivity  
mListener.onArticleSelected(noteUri);  
}  
...  
}

Adding items to theAction Bar（稍后翻译）

控制fragment的生命周期

管理fragment的生命周期和管理activity的生命周期非常相似，像activity一样，fragment也有三种状态。

Resumed ：

在运行的activity中，fragment是可见的。

Paused ：

有另外一个activity在前端，并且可以获得焦点。但是在这个activity上的fragment还是可见的（前端的activity是半透明的或者是没有完全覆盖了下面的activity）；

Stopped ：

这个fragment不可见。宿主activity已经处于stop状态或者这个fragment已经从这个activity中移除，并且存到到了返回栈当中。处于stoped状态的fragment还处于存活状态（所有的状态和成员信息都会保存在系统中）。不管怎样，用户是不可见的，当activity被k之后这个fragment也会被k。

同样和activity一样，你可以通过 Bundle 保存fragment的信息，activity的进程被杀死时并重新创建时，你需要恢复fragment的状态。 你可以在fragment的 onSaveInstanceState()中保存fragment的状态，恢复fragment的装载在 onCreate(),onCreateView(), 或者 onActivityCreated()方法中。

fragment的生命周期与activity的生命周期最大的区别是：他们是如何被存到各自的返回栈中。当一个activity处于stoped状态的时候，默认情况下系统会把activity存放到activity的返回栈中。然而，fragment存放到返回栈中是由宿主activity管理，除非在一个移除fragment事务中你明确调用addToBackStack()。

Figure 3. Theactivity lifecycle's affect on the fragment lifecycle  
配合activity的生命周期

activity的生命周期直接影响其内的fragment的生命周期。比如，activity的生命周期回调方法被调用，就会影响到fragment的相应的fragment的生命周期的回调方法。例如，当activity收到 onPause() ，在activity里面的也将收到fragment。

fragment还有其他一些回调方法，控制它与activity的交互是为了执行类似于创建或者销毁fragment的UI，那些其他的回调方法是：

onAttach()

当fragment已经与activity建立了联系是被调用（这个activity在这里被传递进来）

onCreateView()

创建fragment的相关的视图是被调用

onActivityCreated()

当activity的 onCreate() 方法返回时，被调用。

onDestroyView()

当这个fragment的视图被移除时被调用。

onDetach()

当fragment与activity解除关联时被调用。

fragment的生命周期的执行受宿主activity的直接影响，请看figure3的图解。

一旦activity运行到了resumed 状态，你可以自由的在activity中增加或者移除fragment，因此只有在activity处于resumed状态时，fragment的生命周期才能独立的改变。

然而，当activity不在是resumed状态的时候，fragment又重新被纳入到了这个activity当中去了。  
例子：

[分享](javascript:;)

[举报收藏(0)没用(0)有用(0)](javascript:;)

﻿

[7条评论](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "##)



[fantao](http://my.eoe.cn/578799" \t "_blank)1楼,

这排版…………不谈了

07月12日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[aoratec](http://my.eoe.cn/viver120" \t "_blank)2楼,

哎，博主该排个版面了，全篇没有一个换行，都不“忍心”看了 ......

07月13日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[bingxueer0702](http://my.eoe.cn/873222" \t "_blank)3楼,

文章内容还没看呢，从头到尾一气呵成啊、1

07月15日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[myclover](http://my.eoe.cn/myclover" \t "_blank)4楼,

这排版没法看啊，能不能稍微排版一下呢？

07月15日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[逍遥§№无风](http://my.eoe.cn/myboby" \t "_blank)5楼,

是啊。。这个也太乱了点。。。。

07月15日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[贾小骏](http://my.eoe.cn/morefree" \t "_blank)6楼,

[#1楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/1.html) [@fantao](http://my.eoe.cn/578799" \t "_top)  
[#2楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/2.html) [@aoratec](http://my.eoe.cn/viver120" \t "_top)  
[#3楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/3.html) [@bingxueer0702](http://my.eoe.cn/873222" \t "_top)  
[#4楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/4.html) [@myclover](http://my.eoe.cn/myclover" \t "_top)  
[#5楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/5.html) [@逍遥§№无风](http://my.eoe.cn/myboby" \t "_top)  
  
稍微帮楼主搞了搞

07月15日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)



[逍遥§№无风](http://my.eoe.cn/myboby" \t "_blank)7楼,

[#6楼](http://comment.eoe.cn/index/sort/itemid/5576/model/blog/sort/6.html) [@贾小骏](http://my.eoe.cn/morefree" \t "_top) 这样好多了。楼主的帖子讲的挺全面的。谢了。另外问一下Fragment里面能否使用SharedPreferences ？

07月16日[回复](http://my.eoe.cn/niunaixiaoshu/archive/5576.html" \l "replybody)

窗体顶端



**支持表情，见 [Emoji cheat sheet](http://www.emoji-cheat-sheet.com/" \t "_blank)**

**0/400**

提交回复

窗体底端

[关于我们](http://my.eoe.cn/eoe/page/about.html" \t "_blank) | [招贤纳士](http://my.eoe.cn/eoe/page/job.html" \t "_blank) | [联系我们](http://my.eoe.cn/eoe/page/about.html" \t "_blank) | [eoe安卓客户端](http://a1.eoe.cn/www/app/eoecn_v1.1.0.apk" \t "_blank" \o "eoe安卓客户端) | [客户端源码](https://github.com/eoecn/android-app" \t "_blank" \o "eoe安卓客户端源码) | [在线客服](http://wpa.qq.com/msgrd?V=3&Uin=800090121&Site=eoe%20Android%E5%BC%80%E5%8F%91%E8%80%85%E7%A4%BE%E5%8C%BA&Menu=yes&from=discuz" \t "_blank" \o "在线客服)

eoe致力于构建完善的技术社区, 推荐使用[Chrome](http://www.google.com/chrome/" \t "_blank)浏览本站以获得更佳的视觉体验.

Copyright © 2009 - 2013 eoe.cn All Rights Reversed  
京ICP备11018032号-2 京公网安备110105014978