

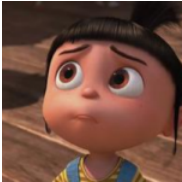
# dreamInTheWorld的专栏

目录视图

摘要视图

RSS

个人资料



dreamInTheWorld

关注

发私信



访问：41065次

积分：1073

等级：

排名：千里之外

原创：65篇

转载：11篇

译文：0篇

评论：24条

文章搜索

文章存档

- 2016年03月 (2)
- 2015年12月 (3)
- 2015年11月 (3)
- 2015年06月 (2)
- 2015年05月 (1)

展开

阅读排行

- 使用setCustomView建立标题... (5998)
- eclipse设置护眼豆沙绿背景色... (3429)
- 安卓Launcher之获取手机安装... (1330)
- Andorid开源框架greenDAO (1162)
- 模式对话框PopupWindow实现... (1115)
- Fragment之间的通信 (941)
- Fragment实际应用中的生命周期 (888)
- 小米手机一些系统应用 (796)
- 安卓开发中通过图片的绝对路... (789)
- 使用AudioManager设置Androi... (755)

## Fragment实际应用中的生命周期

标签：Fragment ViewPager android应用 FragmentManager

2014-08-27 16:12

888人阅读

评论(0)

收藏

举报

分类：Android进阶 (19)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

## 1 Fragment加载到Activity中的生命周期

在Fragment中仍然能看到Activity的生命周期，如果只是纯粹的切换，Fragment的生命周期还是存在一定的意义，下面解析一下。

因为onCreate()--onStart()--onResume()是一块。如果是从另外的Fragment进入，则会在onCreate()--onStart()之间先执行上一个Fragment的销毁动作onPause()--onStop()。然后紧接着才是onStart()--onResume()

相对于Activity而言，周期简洁，onResume()就是可以被用户看到的。但是到了Fragment中，虽然还有这个周期，不过可以已经完全不在是纯粹的“被用户看到”。

保留了onResume()这个方法，不过已经没有很大意义。我们可以调用到另外两个API来知道Fragment生命周期中他自己的onResume()。那就是

onHiddenChanged(boolean)

setVisible(boolean)

这两个重要的API，会在下面“事务处理中的生命周期”和“ViewPager使用Fragment”两块中进行解析。

## 2 在beginTransaction()事务处理中的生命周期

```
[java]

01. public class MainActivity extends FragmentActivity {
02.
03.     private FragmentManager fmanager;
04.     private Button button;
05.     private Button button2;
06.     private Fragment f1;
07.     private Fragment f2;
08.     private String tag = "MainActivity";
09.
10.     @Override
11.     protected void onCreate(Bundle savedInstanceState) {
12.         super.onCreate(savedInstanceState);
13.         setContentView(R.layout.activity_main);
14.         button = (Button) findViewById(R.id.button);
15.         button2 = (Button) findViewById(R.id.button2);
16.         fmanager = getSupportFragmentManager();
17.         f1 = new FragmentAdemo("#f1");
18.         f2 = new FragmentAdemo("#f2");
19.         Log.i(tag, tag+"#new完成");
20.         fmanager.beginTransaction().add(R.id.layout,f1).commit();
21.         fmanager.beginTransaction().add(R.id.layout,f2).commit();
22.
23.
24.         button.setOnClickListener(new View.OnClickListener() {
25.
26.             @Override
27.             public void onClick(View v) {
28.                 fmanager.beginTransaction().show(f1).commit();
29.                 fmanager.beginTransaction().hide(f2).commit();
30.             }
31.         });
32.
33.         button2.setOnClickListener(new View.OnClickListener() {
34.             @Override
35.             public void onClick(View v) {
36.                 fmanager.beginTransaction().show(f2).commit();
37.                 fmanager.beginTransaction().hide(f1).commit();
38.             }
39.         });
40.     }
41. }
```

FragmentAdemo的实现就是普通的Fragment，就不贴出代码了。

启动运行上面代码可以看到这样的输出

```
#f1#onAttach()
#f1#onCreate()
#f1#onCreateView()
#f1#onViewCreated()
#f1#onActivityCreated()
#f2#onAttach()
#f2#onCreate()
#f2#onCreateView()
#f2#onViewCreated()
#f2#onActivityCreated()
```

如果注释掉其中的f2，并且添加完成所以的生命周期，运行结果是这样的：

```
MainActivity#new完成
#f1#onAttach()
#f1#onCreate()
#f1#onCreateView()
#f1#onViewCreated()
#f1#onActivityCreated()
#f1#onStart()
#f1#onResume()
#f1#####返回键被摁下###
#f1#onPause()
#f1#onStop()
#f1#onDestroyView()
#f1#onDestroy()
#f1#onDetach()
```

这个结果跟<http://www.cnblogs.com/mengdd/archive/2013/01/08/2851368.html>中讲解的生命周期一致。

下面执行Button按键的，相互交替摁下，是下面的结果：

```
#f1#onCreateView()
#f2#onCreateView()
#f2#onHiddenChanged=true
#f2#onHiddenChanged=false
#f1#onHiddenChanged=true
#f1#onHiddenChanged=false
#f2#onHiddenChanged=true
#f2#onHiddenChanged=false
#f1#onHiddenChanged=true
```

嗯，没错，生命周期内没有起反应了，说明，视图没有被销毁，这样的话，就实现了视图保存。

同时调用到了生命周期内的**onHiddenChanged(boolean)**用来区别是不是被显示到前台。

可以使用replace()函数代替hide()和show()共同工作的效果，但是生命周期就不一样了：

```
[java]

01. button.setOnClickListener(new View.OnClickListener() {
02.
03.     @Override
04.     public void onClick(View v) {
05.         fmanager.beginTransaction().replace(R.id.layout, f1).commit();
06.     }
07. });
08.
09. button2.setOnClickListener(new View.OnClickListener() {
10.
11.     @Override
12.     public void onClick(View v) {
13.         fmanager.beginTransaction().replace(R.id.layout, f2).commit();
14.     }
15. });
```

可以看到的生命周期是这样的：

#f1#onCreateView()	
#f2#onCreateView()	
#f2#onPause()	button摁下
#f2#onStop()	
#f2#onDestroy()	
#f1#onPause()	buton2摁下
#f1#onStop()	
#f1#onDestroy()	
#f2#onCreateView()	button摁下
#f2#onPause()	
#f2#onStop()	
#f2#onDestroy()	
#f1#onCreateView()	

这样，可以看到replace()很暴力，直接就销毁上一家。

同时没有生命周期onHiddenChanged(boolean)。

处于保存视图，个人提倡使用hide-show组合。

### 3 在ViewPager中的生命周期

VPagerActivity.java

```
[java]

01. import android.support.v4.app.FragmentActivity;
02. import android.support.v4.app.FragmentManager;
```

```
03. import android.support.v4.view.ViewPager;
04.
05. public class VPagerActivity extends FragmentActivity {
06.
07.     private ViewPager pager;
08.     private DemoAdapter adapter;
09.
10.     @Override
11.     protected void onCreate(Bundle savedInstanceState) {
12.         super.onCreate(savedInstanceState);
13.         setContentView(R.layout.activity_vpager);
14.         pager = (ViewPager) findViewById(R.id.viewPager);
15.         FragmentManager fm;
16.         fm = getSupportFragmentManager();
17.         adapter = new DemoAdapter(fm, this);
18.         pager.setAdapter(adapter);
19.     }
20. }
```

布局文件：

activity\_vpager.xml

```
[html]
01. <?xml version="1.0" encoding="utf-8"?>
02. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03.     android:layout_width="match_parent"
04.     android:layout_height="match_parent" >
05.
06.     <android.support.v4.view.ViewPager
07.         android:id="@+id/viewPager"
08.         android:layout_width="match_parent"
09.         android:layout_height="match_parent"
10.     />
11.
12. </RelativeLayout>
```

适配器

DemoAdapter.java

```
[java]
01. import android.support.v4.app.Fragment;
02. import android.support.v4.app.FragmentManager;
03. import android.support.v4.app.FragmentPagerAdapter;
04.
05. public class DemoAdapter extends FragmentPagerAdapter {
06.
07.     Context context;
08.     private ArrayList<Fragment> datas;
09.
10.     public DemoAdapter(FragmentManager fm, Context context) {
11.         super(fm);
12.         this.context = context;
13.         datas = new ArrayList<Fragment>();
14.         for(int i=1;i<=5;i++){
15.             datas.add(new Fragment1("#F[00"+i+"]"));
16.         }
17.     }
18.
19.     @Override
20.     public Fragment getItem(int index) {
21.         return datas.get(index);
22.     }
23.
24.     @Override
25.     public int getCount() {
26.         return datas.size();
27.     }
28.
29. }
```

ViewPager中的Fragment生命周期运行效果：

#F[001]#setUserVisibleHint=false #F[002]#setUserVisibleHint=false #F[001]#setUserVisibleHint=true #F[001]#onCreateView #F[002]#onCreateView	开始进入第一页
#F[003]#setUserVisibleHint=false #F[001]#setUserVisibleHint=false #F[002]#setUserVisibleHint=true #F[003]#onCreateView	进入第二页
#F[004]#setUserVisibleHint=false #F[002]#setUserVisibleHint=false #F[003]#setUserVisibleHint=true #F[001]#onPause() #F[001]#onStop() #F[004]#onCreateView	进入第三页
#F[005]#setUserVisibleHint=false #F[003]#setUserVisibleHint=false #F[004]#setUserVisibleHint=true #F[002]#onPause() #F[002]#onStop() #F[005]#onCreateView	进入第四页
#F[004]#setUserVisibleHint=false #F[005]#setUserVisibleHint=true #F[003]#onPause() #F[003]#onStop()	进入第五页
#F[003]#setUserVisibleHint=false #F[005]#setUserVisibleHint=false #F[004]#setUserVisibleHint=true #F[003]#onCreateView	回到第四页
#F[002]#setUserVisibleHint=false #F[004]#setUserVisibleHint=false #F[003]#setUserVisibleHint=true #F[002]#onCreateView #F[005]#onPause() #F[005]#onStop()	回到第三页

可以看到，只要不是关联的左右页，都会被pause和stop掉。

然后再次进入调用onCreateView()重新执行生命周期。

但是onCreateView()只是预加载的，所以要知道是不是正在被用户看到，这个时候：

需要用**setUserVisibleHint()**来进行判断。但是刚进入的时候，会执行两下setUserVisibleHint()，所以需要注意。

如果第一次进入的不是第一页，也是会激活旁边的两个Fragment。

关于ViewPager还会出现一种情况，就是使用setCurrentItem(int)进行调用的时候，就会激活到沿路的所有Fragment页面

#F[001]#setUserVisibleHint=false #F[002]#setUserVisibleHint=false #F[001]#setUserVisibleHint=true #F[001]#onCreateView #F[002]#onCreateView	第一次进入
#F[004]#setUserVisibleHint=false #F[003]#setUserVisibleHint=false #F[005]#setUserVisibleHint=false #F[001]#setUserVisibleHint=false #F[004]#setUserVisibleHint=true #F[004]#onCreateView #F[003]#onCreateView #F[005]#onCreateView #F[002]#onPause() #F[002]#onStop() #F[001]#onPause() #F[001]#onStop()	setCurrentItem(3)

虽然沿路激活了一部分，但是因为没有onStart所以onStop也是不会出现的。

本文来自CSDN博客 转载请联系作者并且标明出处<http://blog.csdn.net/dreamintheworld/article/details/38872249>

顶

0

踩

0

- [上一篇](#) 线程中阻塞通信Condition的wait()和signal()
- [下一篇](#) Fragment之间的通信

我的同类文章

Android进阶（19）

• 重写InputMethodService编写一个简单...	2016-03-17	阅读 106	• 键盘控件KeyboardView的使用	2016-03-17	阅读 95
• 通过Runtime.getRuntime().exec使用Linu...	2015-11-20	阅读 154	• 使用WifiManager和ScanResult进行Wifi...	2015-04-15	阅读 655
• 使用AlarmManager实现定时功能	2015-03-31	阅读 258	• 最直观的讲一下安卓动画在四个最常用...	2015-03-02	阅读 335
• Android日志服务例子，将日志记录在...	2015-02-26	阅读 276	• 安卓3.2后屏幕适配阶段性总结	2015-01-22	阅读 512
• 菜鸟来解释线程中的循环者Looper读取...	2015-01-19	阅读 364	• 使用AudioManager设置Android系统的...	2014-11-19	阅读 751
• Android 通过应用设置系统日期和时间...	2014-10-31	阅读 410			
更多立音					

猜你在找

【安卓开发实战免费微课】第3罐...	Fragment
反编译Android应用	Android 应用开发GitHub 优秀的 ...
Android应用的调试	Android项目Tab类型主界面大总...
Android应用更新实现策略	Android Fragment 真正的完全解...
Android应用开发流程及友盟统计...	Android Fragment 真正的完全解...

查看评论

暂无评论

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack	VPN	Spark	ERP	IE10
Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	BI	HTML5	Spring	Apache	.NET	API	HTML
SDK	IIS	Fedora	XML	LBS	Unity	Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra	
CloudStack	FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo	Compuware	大数据		
aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr	Angular	Cloud Foundry	Redis	Scala	Django	
Bootstrap														

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服      杂志客服      微博客服      webmaster@csdn.net      400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

