## 1.悬浮按钮 FloatingActionButton

FloatingActionButton改变icon颜色用android:tint:”@color/.....”

FloatingActionButton改变background颜色用app:backgroundTint:”@color/.....”

<android.support.design.widget.FloatingActionButton

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/ic\_add"

app:backgroundTint="@color/orange"

app:borderWidth="0dp"

app:elevation="6dp"

app:fabSize="normal" >

设置<LinearLayout>标签的 android:showDividers属性可以显示分隔线。

如果有多个LinearLayout，显示效果和在 LinearLayout之间加分隔线是一样的。

android:showDividers属性可以设置如下4个值：

none：不显示分隔线；

beginning：在LinearLayout的开始处显示分隔线；

end：在Linearlayout的结尾处显示分隔线；

middle：在LinearLayout中的每两个组件间显示分隔线。

除了需要设置android:showDividers属性外，还要设置android:divider属性，该属性表示分隔线的图像。

## AdapterView

setOnItemClickListner\setOnItemSelectedListener 可以为其添加点击事件

listview.setOnItemClickListener\selected (new OnItemClickListener)

@Override

//ItemClick Public void onItemClick(AdapterView<?> parent, View view , int position, long id)

{

Shuchu(name[position])

}

注意使用select的时候要同时重写 onNotingSelected 方法

## ListView divider footer\headerDividerHeight

4.ArrayAdapter

ArrayAdapter <String> name = new ArrayAdapter <String >(Context this,textViewResource R.layout.textview, shuju list/array)

## 5.二维数组

Private String [] []arms = nnew String [] []{ {“s”,”a”,”d”},{ “idj”,”doah”,”territory” } } ;

## fragment的数据保存与恢复

单个Fragment遭遇突发情况

仍然是用以下突发情况进行测试：

点击back键

点击锁屏键

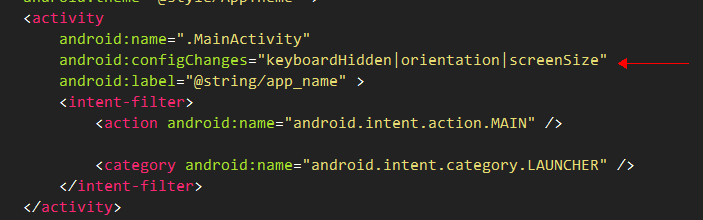
点击home键

其他APP进入前台

启动了另一个Activity

屏幕方向旋转

APP被Kill



这么做的目的是当屏幕方向发生改变的时候，fragment所依附的Activity并不会重新销毁再创建，让****情况相对简单一点****。

Fragment回到返回栈

1.临时数据 对于临时数据，我们使用onSaveInstanceState方法进行保存，并且在onCreateView方法中恢复（请注意是onCreateView）。

2.永久数据 对于持久性数据，我们要在onPause方法中进行存储。

对比这两张生命周期方法的图，能得出两个结论。  
　　1.无论任务栈中fragment数量为多少，onSaveInstanceState方法都没有调用  
　　2.当fragment任务栈中有多个fragment时，进入下一个fragment时，并不会销毁fragment实例，而是仅仅销毁视图，最终调用的方法为onDestoryView。  
　　所以此时我们要去保存临时数据，并不能仅保存在onSaveInstanceState中（因为它可能不会调用），还应该在onDestoryView方法中进行保存临时数据的操作，源码如下：



结束之前我们来一句话总结下本文：  
Fragment对临时数据的保存，仅仅依靠onSaveInstanceState方法是不行的，还需要在onDestoryView中进行相应操作，具体参考上面的代码。

Fragment中对于一些持久性的数据，仍应在onPause中保存。

## Activity数据保存与恢复

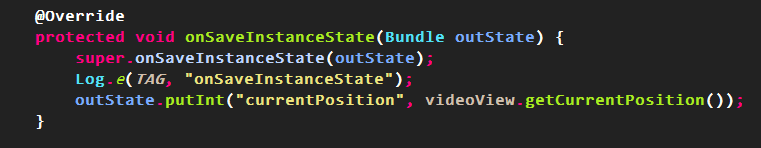
在我写的这个demo中，onSaveInstanceState的调用是处于onPause和onStop之间的，（下面关于Activity的生命周期方法，会讲解一些值得大家注意的），我查阅了一下资料，能保证的是onSaveInstanceState方法会在onStop之前调用，但是是否在onPause之前就不一定了。

结论： google工程师们对onSaveInstanceState如此设计就是让其完成对一些临时的、非永久数据存储并进行恢复。什么样的数据属于临时数据呢？举个例子，比如EditText中输入的内容，CheckBox是否勾选，ScrollView的滑动位置，目前视频的播放位置等等。

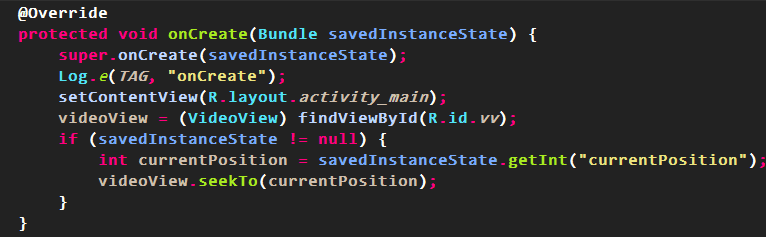
（TextView）其他View控件都有相似的实现原理（自动保存）。值得一提的是，只有当你给这个wiget在xml中指定id时，它才具有保存数据并且恢复的能力，并且不同的wiget还不能共用这个id，否则会出现数据覆盖的情况。

### onSaveInstanceState的使用姿势

比如我们要保存当前视频的播放进度，这个显然控件没有帮我们实现onSaveInstanceState，所以就只能靠自己了，代码如下所示。



保存临时数据



恢复临时数据

当在onCreate取出临时数据时，记得加一个非空判断。

看到这里，也许你认为本文该就此结束了，不过在回过头看看，我们刚才一直强调的是临时数据，毕竟onSaveInstanceState本身就是为临时数据服务的，但是一些永久性质的数据，比如插入数据库的操作，我们应该在什么方法中对其进行保存呢？

## ? : ;的新用法（i>0?i++:j=2）

**public** **static** **boolean** isNotNull(String txt){

**return** txt!=**null** && txt.trim().length()>0 ? **true**: **false**;

    }

## 9. chronometer 用于计时的实现

- bug1：当点击计时过一段时间再点击结束，然后在点击开始在点击结束，并不会继续第一次计时，而是返回新的时间。

- bug2：还是点击计时后在点击结束，再点击计时会出现乱码的现象，但是并不影响计时的时间。

在fragment上实现的：

**import** android.os.Bundle;  
**import** android.os.SystemClock;  
**import** android.support.annotation.Nullable;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.Button;  
**import** android.widget.Chronometer;  
**import** android.widget.ImageButton;  
**import** android.widget.TextView;  
**import** android.widget.Toast;

**public class** FragmentTimer **extends** android.support.v4.app.Fragment {  
  
 **static long** *start\_time* = 0 ;  
 **static long** *end\_time* = 0 ;  
 **static long** *mRecordTime* = 0;  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.***fragment\_timer***,container,**false**);  
 **final** ImageButton startButton = (ImageButton)view.findViewById(R.id.***start***);  
 **final** ImageButton stopButton = (ImageButton)view.findViewById(R.id.***stop***);  
 **final** Chronometer chronometer = (Chronometer)view.findViewById(R.id.***timer***);  
 **final** Button submit = (Button)view.findViewById(R.id.***submit***);  
 **final** TextView textTime = (TextView)view.findViewById(R.id.***learntime***) ;  
 submit.setEnabled(**false**);  
  
  
  
  
 startButton.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 **if** (*mRecordTime* != 0) {  
 chronometer.setBase(chronometer.getBase() + (SystemClock.*elapsedRealtime*() - *mRecordTime*));  
 } **else** {  
 chronometer.setBase(SystemClock.*elapsedRealtime*());  
 }  
  
 *start\_time* = SystemClock.*elapsedRealtime*();  
 chronometer.start();  
 startButton.setEnabled(**false**);  
 stopButton.setEnabled(**true**);  
 submit.setEnabled(**false**);  
 textTime.setText(**"正在计时"**);  
  
 }  
 });  
 stopButton.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 chronometer.stop();  
 *mRecordTime* = SystemClock.*elapsedRealtime*();  
 *end\_time* = chronometer.getBase();  
 startButton.setEnabled(**true**);  
 stopButton.setEnabled(**false**);  
 submit.setEnabled(**true**);  
 textTime.setText(**"计时停止"**);  
 }  
 });  
  
 submit.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **long** time1 = SystemClock.*elapsedRealtime*() - *start\_time*;  
 **long** time2 = (time1/1000)/59;  
 **long** time = (time1/1000)%59;  
 textTime.setText(**"您学习了"**+time2+**"分"**+time+**"秒"**);  
 chronometer.setBase(SystemClock.*elapsedRealtime*());  
 chronometer.stop();  
 Toast.*makeText*(getActivity(),**"您的数据已提交"**,Toast.***LENGTH\_SHORT***).show();  
 submit.setEnabled(**false**);  
 }  
 });  
  
  
 **return** view;  
 }  
  
}

相应的Xml：

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:id="@+id/fragmenttimer" android:layout\_height="match\_parent"**>  
  
  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/start"  
 android:layout\_gravity="center"  
 android:layout\_weight="1"  
 android:background="@drawable/start"  
 android:layout\_marginBottom="29dp"  
 android:layout\_above="@+id/timer"  
 android:layout\_toStartOf="@+id/timer"** />  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:layout\_gravity="center"  
 android:id="@+id/stop"  
 android:background="@drawable/stop"  
 android:layout\_alignTop="@+id/start"  
 android:layout\_toRightOf="@+id/timer"  
 android:layout\_toEndOf="@+id/timer"** />  
  
 <**Chronometer  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/timer"  
 android:textSize="50dp"  
 android:layout\_centerVertical="true"  
 android:layout\_centerHorizontal="true"** />  
  
 <**Button  
 android:id="@+id/submit"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="59dp"  
 android:text="提交"  
 android:layout\_below="@+id/timer"  
 android:layout\_centerHorizontal="true"** />  
  
 <**TextView  
 android:id="@+id/learntime"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:layout\_marginTop="27dp"  
 android:layout\_below="@+id/timer"  
 android:layout\_centerHorizontal="true"** />  
  
</**RelativeLayout**>

# ToolBar

|  |  |
| --- | --- |
| 4down voteaccepted | Define a Menu for your Toolbar in the *res/menu* resource folder, for example:  *toolbar\_menu.xml*  <?xml version="1.0" encoding="utf-8"?><menu  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  tools:context=".activity.MainActivity">  <item  android:id="@+id/action\_sign\_out"  android:title="@string/toolbar\_sign\_out"  app:showAsAction="never"/></menu>  Setting app:showAsAction="never" ensures that this MenuItem will not be shown in the Toolbar, but placed in the overflow menu instead.  The theme of your Activity should be (or derive from) one of the NoActionBar AppCompat themes (Theme.AppCompat.Light.NoActionBar for example).  In your Activity, set up your Toolbar:  Toolbar myToolbar = (Toolbar) findViewById(R.id.my\_toolbar);  setSupportActionBar(myToolbar);  And override onCreateOptionsMenu():  @Override public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.toolbar\_menu, menu);  return true; }  You can override onOptionsItemSelected() to define the onClick behaviour of your MenuItem(s):  @Override public boolean onOptionsItemSelected(MenuItem item) {  switch (item.getItemId()) {  case R.id.action\_sign\_out: {  // do your sign-out stuff  break;  }  // case blocks for other MenuItems (if any)  }  return false; } |

# **【Android】ViewPager深入解析（一）**

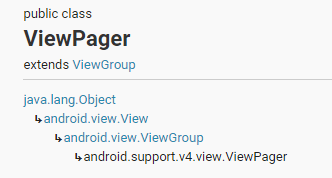
2015-12-07 11:59:2816519浏览[19评论](http://www.imooc.com/article/2580" \l "comment)

话说小伙伴们在使用App的时候有没有注意到很多App的首页都是可以左右滑动的页面呢？很多App还有绚丽的轮播图广告。那么如何实现这样的效果呢？相信很多小伙伴都知道可以用ViewPager来完成！不过也许很多小伙伴也跟皮卡丘一样，虽然一直在使用ViewPager，但是大部分时间都是在ctrl+c、ctrl+v~~~  
  
但是，很多小伙伴也许并没有详细地了解过ViewPager，所谓“知其然知其所以然”，那么今天皮卡丘就和大家一起来学习下ViewPager吧~~~

每个小标题后面的句子，素材来自于一款网络游戏，大家随便看看就可以了，跟主题无关，我只是觉得好玩......

****1、江湖相逢****

*犹记与你初逢扬州湖畔，你只回眸一笑，我却惊艳了时光。*

关于ViewPager，我们先来看看api中的继承关系  
  
从图里可以看出，ViewPager继承自ViewGroup，也就是ViewPager是一个容器类，可以包含其他的View类。然后我们在看看api中的定义：

Layout manager that allows the user to flip left and right through pages of data. You supply an implementation of a PagerAdapter to generate the pages that the view shows.

Note this class is currently under early design and development. The API will likely change in later updates of the compatibility library, requiring changes to the source code of apps when they are compiled against the newer version.

ViewPager is most often used in conjunction with Fragment, which is a convenient way to supply and manage the lifecycle of each page. There are standard adapters implemented for using fragments with the ViewPager, which cover the most common use cases. These are FragmentPagerAdapter and FragmentStatePagerAdapter; each of these classes have simple code showing how to build a full user interface with them.

有一些英语没有学好的小伙伴可能不是很能理解这一段描述。快去跟你们的英语老师道歉啊！嘛，这里皮卡丘就先给你们进行一下翻译吧，不过因为水平有限，可能会有一些翻译错误，如果大家发现错误的话欢迎留言哦。

ViewPager是一个允许使用者左右滑动数据页面的布局管理器。你可以通过一个适配器（PagerAdapter）来管理要显示的页面。

不过要注意的是，这个类目前还处于初期的设计和开发。随着今后兼容库的更新，API文档也会进行更改，同时应用程序在编译时也需要对代码进行一定的修改。

ViewPager更多的时候会与Fragment一起使用，这是一种很好的方法来管理各个页面的生命周期。Android提供了一些专门的适配器来让ViewPager与Fragment一起工作，也就是FragmentPagerAdapter与FragmentStatePagerAdapter。他们基本上可以满足大部分常见的永续需求，并且他们都有简单的代码样例来展示如何用他们来建立一个完整的用户页面。

通过这一段描述，大家应该对ViewPager有了一个大致的了解，皮卡丘大致把常用的信息再总结一下：

1、ViewPager主要用来左右滑动。（类似图片轮播）  
2、ViewPager要用适配器来连接“视图”和“数据”。（大家可以联想下listview的使用方法，原理是类似的）  
3、官方推荐ViewPager与Fragment一起使用，并且有专门的适配器。

好啦，看到这里，小伙伴们是不是对这么一个神奇的组件充满了兴趣呢？那我们赶紧来写一个最简单的ViewPager吧。

****2、相见恨晚****

*大家都说我逍遥此身君子意，但与你把酒言欢之时，我却知道，逍遥的心亦有了牵挂。所谓相见恨晚，大概就是如此吧。*

在创建项目之前，我们先大致来描述下该项目的功能：viewpager中包含3个视图，并且这3个视图可以通过左右滑动来进行切换。很简便吧？我们先创建一个ViewPagerDemo的工程。  
既然我们需要包含3个视图，我们就先准备3个页面的布局吧（很简单的布局，就是居中显示一个图片）：  
page1.xml

<?xml version="1.0" encoding="utf-8"?><RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:src="@drawable/view1" />

</RelativeLayout>

page2.xml

<?xml version="1.0" encoding="utf-8"?><RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:src="@drawable/view2" />

</RelativeLayout>

page3.xml

<?xml version="1.0" encoding="utf-8"?><RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:src="@drawable/view3" />

</RelativeLayout>

然后是我们的主页面布局，只有一个ViewPager：  
activity\_main.xml

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

>

<android.support.v4.view.ViewPager

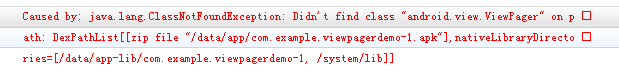
android:id="@+id/myViewPager"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

></android.support.v4.view.ViewPager>

</LinearLayout>

这里需要注意的是，ViewPager引入的时候必须写完整android.support.v4.view.ViewPager，如果我们按照一般控件那样直接写ViewPager，就会报以下的错误  
  
其实从报错描述中我们大致能看出来是因为无法在android.view下面找到ViewPager这个类，这里的具体原因涉及到兼容包的问题，如果有机会的话皮卡丘也会以后再分享，不过这篇文章就先不深入研究啦。

好啦，我们已经把该准备的布局文件都准备好了，接下来我们开始来敲代码。  
从API中我们看出来，viewpager是通过适配器来进行管理的（数据源---适配器---视图）。我们分别来看看：  
****数据源：****这里的数据比较简单，就是包含3个View的一个list

private View page1, page2, page3; // ViewPager包含的页面private List<View> pageList; // ViewPager包含的页面列表，一般给adapter传的是一个list

MainActivity.java......

LayoutInflater inflater = getLayoutInflater();

page1 = inflater.inflate(R.layout.page1, null);

page2 = inflater.inflate(R.layout.page2, null);

page3 = inflater.inflate(R.layout.page3, null);

pageList = new ArrayList<View>();

pageList.add(page1);

pageList.add(page2);

pageList.add(page3);

****视图：****一般视图都是比较简单的，这里也不例外，就是一个ViewPager

private ViewPager myViewPager; // 要使用的ViewPager

......

myViewPager = (ViewPager) findViewById(R.id.myViewPager);

****适配器：****在大多数使用适配器的控件里，适配器相对于数据源和视图来说都更加复杂，同时也决定了这个控件主要的功能。ViewPager也不例外，所以我们有必要对PagerAdapter深入了解一下。  
我们先来看看API中对PagerAdapter的描述：

Base class providing the adapter to populate pages inside of a ViewPager. You will most likely want to use a more specific implementation of this, such as FragmentPagerAdapter or FragmentStatePagerAdapter.

When you implement a PagerAdapter, you must override the following methods at minimum:

* instantiateItem(ViewGroup, int)
  + destroyItem(ViewGroup, int, Object)
  + getCount()
  + isViewFromObject(View, Object)

PagerAdapter is more general than the adapters used for AdapterViews. Instead of providing a View recycling mechanism directly ViewPager uses callbacks to indicate the steps taken during an update. A PagerAdapter may implement a form of View recycling if desired or use a more sophisticated method of managing page Views such as Fragment transactions where each page is represented by its own Fragment.

ViewPager associates each page with a key Object instead of working with Views directly. This key is used to track and uniquely identify a given page independent of its position in the adapter. A call to the PagerAdapter method startUpdate(ViewGroup) indicates that the contents of the ViewPager are about to change. One or more calls to instantiateItem(ViewGroup, int) and/or destroyItem(ViewGroup, int, Object) will follow, and the end of an update will be signaled by a call to finishUpdate(ViewGroup). By the time finishUpdate returns the views associated with the key objects returned by instantiateItem should be added to the parent ViewGroup passed to these methods and the views associated with the keys passed to destroyItem should be removed. The method isViewFromObject(View, Object) identifies whether a page View is associated with a given key object.

A very simple PagerAdapter may choose to use the page Views themselves as key objects, returning them from instantiateItem(ViewGroup, int) after creation and adding them to the parent ViewGroup. A matching destroyItem(ViewGroup, int, Object) implementation would remove the View from the parent ViewGroup and isViewFromObject(View, Object) could be implemented as return view == object;.

PagerAdapter supports data set changes. Data set changes must occur on the main thread and must end with a call to notifyDataSetChanged() similar to AdapterView adapters derived from BaseAdapter. A data set change may involve pages being added, removed, or changing position. The ViewPager will keep the current page active provided the adapter implements the method getItemPosition(Object).

又到了皮卡丘的英语小讲堂啦，依然是大致的翻译：

PagerAdapter是用于“将多个页面填充到ViewPager”的适配器的一个基类，大多数情况下，你们可能更倾向于使用一个实现了PagerAdapter并且更加具体的适配器，例如FragmentPagerAdapter或者FragmentStatePagerAdapter。

当你实现一个PagerAdapter时，你至少需要重写下面的几个方法：

* instantiateItem(ViewGroup, int)
  + destroyItem(ViewGroup, int, Object)
  + getCount()
  + isViewFromObject(View, Object)

PagerAdapter比很多AdapterView的适配器更加通用。ViewPager使用回调机制来显示一个更新步骤，而不是直接使用视图回收机制。如果需要时，PagerAdapter也可以实现视图回收方法，或者直接使用一种更加巧妙的方法来管理页面，比如直接使用能够管理自身事务的Fragment。

ViewPager并不直接管理页面，而是通过一个key将每个页面联系起来。这个key用来跟踪和唯一标识一个给定的页面，且该key独立于adapter之外。PagerAdapter中的startUpdate(ViewGroup)方法一旦被执行，就说明ViewPager的内容即将开始改变。紧接着，instantiateItem(ViewGroup, int)和/或destroyItem(ViewGroup, int, Object)方法将会被执行，然后finishUpdate(ViewGroup)的执行就意味着这一次刷新的完成。当finishUpdate(ViewGroup)方法执行完时，与instantiateItem(ViewGroup, int)方法返回的key相对应的视图将会被加入到父ViewGroup中，而与传递给destroyItem(ViewGroup, int, Object)方法的key相对应的视图将会被移除。isViewFromObject(View, Object)方法则判断一个视图是否与一个给定的key相对应。

一个简单的PagerAdapter会选择将视图本身作为key，在将视图创建并加入父ViewGroup之后通过instantiateItem(ViewGroup, int)返回。这种情况下，destroyItem(ViewGroup, int, Object) 的实现方法只需要将View从ViewGroup中移除即可，而isViewFromObject(View, Object)的实现方法可以直接写成return view == object;。

PagerAdapter支持数据集的改变。数据集的改变必须放在主线程中，并且在结束时调用notifyDataSetChanged()方法，这与通过BaseAdapter适配的AdapterView类似。一个数据集的改变包含了页面的添加、移除或者位移。ViewPager可以通过在适配器中实现getItemPosition(Object)方法来保持当前页面处于运行状态。

呼，看完了这一段描述，小伙伴们是不是有点累了呢~~~不过同时也一定对ViewPager的适配器有了更多的了解了吧？我们来筛选下初学时常用的知识点（如果要深入了解的话，还是要把其他的知识也好好掌握哦）：

* instantiateItem(ViewGroup, int)负责初始化指定位置的页面，并且需要返回当前页面本身（其实不一定要View本身，只要是能唯一标识该页面的key都可以，不过初学者一般就先用View本身作为key就可以啦）；
* destroyItem(ViewGroup, int, Object)负责移除指定位置的页面；
* isViewFromObject(View, Object)里直接写“return view == object;”即可（当然，如果你在instantiateItem(ViewGroup, int)里返回的不是View本身，那就不能这么写哦）；
* 在描述中并未提及到getCount()方法，不过这个比较简单，也很常见，就是返回要展示的页面数量。

看了这么多理论知识，小伙伴们有没有想立刻动手写一下代码呢？于是我们回到刚才的项目，开始写一个适配器。我们新建一个类MyPagerAdapter。

MyPagerAdapter.java

package com.example.viewpagerdemo;

import java.util.List;

import android.support.v4.view.PagerAdapter;import android.view.View;import android.view.ViewGroup;

public class MyPagerAdapter extends PagerAdapter {

private List<View> pageList;

public MyPagerAdapter(List<View> pageList) {

this.pageList = pageList;

}

@Override

public int getCount() {

// TODO Auto-generated method stub

// 返回要展示的图片数量

return pageList.size();

}

@Override

public boolean isViewFromObject(View arg0, Object arg1) {

// TODO Auto-generated method stub

// 刚开始用viewpager就直接写“return arg0 == arg1;”就好啦

return arg0 == arg1;

}

......

@Override

public void destroyItem(ViewGroup container, int position, Object object) {

// TODO Auto-generated method stub

// 将当前位置的View移除

container.removeView(pageList.get(position));

}}

结合之前的翻译，相信小伙伴们都能比较好地理解这一段代码吧。接下来就简单了，和所有的适配器一样，将viewpager与适配器绑定就可以了。

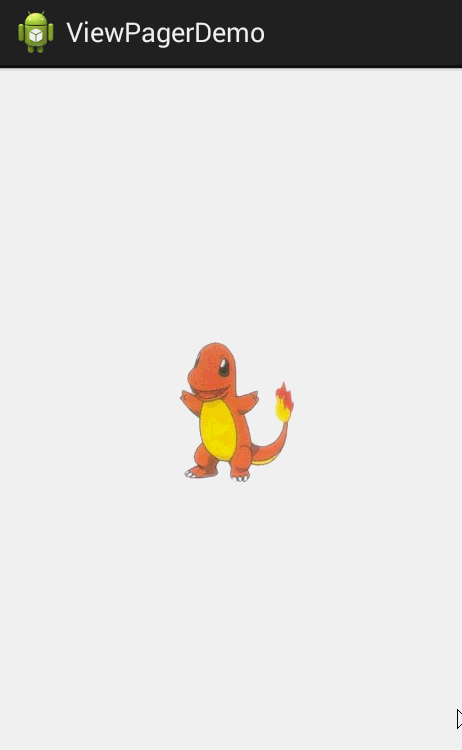
MainActivity.java

private MyPagerAdapter myPagerAdapter; // 适配器

......

myPagerAdapter = new MyPagerAdapter(pageList);

myViewPager.setAdapter(myPagerAdapter);

好啦，是时候运行程序了，我们来看看效果吧~~~  


哈哈哈哈哈哈哈哈哈哈是不是很好玩，当然篇幅所限，这只是ViewPager最最简单的功能，接下来皮卡丘还会继续写ViewPager的一些更高级的用法，欢迎小伙伴们继续支持~~~

  
皮卡~皮卡丘~~~

接下来贴出本次demo的完整代码（慕课网暂时不能上传压缩包，所以先把代码都贴在文章里啦，不过其实建议是不要看demo，直接按照文章自己敲是最好哒）  
MainActivity.java

package com.example.viewpagerdemo;

import java.util.ArrayList;import java.util.List;

import android.os.Bundle;import android.app.Activity;import android.support.v4.view.ViewPager;import android.view.LayoutInflater;import android.view.View;

public class MainActivity extends Activity {

private ViewPager myViewPager; // 要使用的ViewPager

private View page1, page2, page3; // ViewPager包含的页面

private List<View> pageList; // ViewPager包含的页面列表，一般给adapter传的是一个list

private MyPagerAdapter myPagerAdapter; // 适配器

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

initView();

}

private void initView() {

// TODO Auto-generated method stub

myViewPager = (ViewPager) findViewById(R.id.myViewPager);

LayoutInflater inflater = getLayoutInflater();

page1 = inflater.inflate(R.layout.page1, null);

page2 = inflater.inflate(R.layout.page2, null);

page3 = inflater.inflate(R.layout.page3, null);

pageList = new ArrayList<View>();

pageList.add(page1);

pageList.add(page2);

pageList.add(page3);

myPagerAdapter = new MyPagerAdapter(pageList);

myViewPager.setAdapter(myPagerAdapter);

}

}

[](http://www.imooc.com/u/2200734/articles)[snsd](http://www.imooc.com/u/2200734/articles" \t "http://www.imooc.com/article/_blank)16F

皮卡丘老师忘记重写这个方法了 @Override public Object instantiateItem(ViewGroup container, int position) { //每次滑动的时候把视图添加到viewpager container.addView(pageList.get(position)); return pageList.get(position); }

# 自定义控件

Public class LableEditText extends LinearLayout

{

Private TextView txv;

Private String lableText;

Private int lableFontSize;

Private String lablePosition;

Public lableEditText (Context context,AttributeSet attrs)

{

Super(context,attrs);

//读取labletext属性的资源id

Int resourceId = attrs.getAttributeResourceValue(null,”lableText”,0);

//未获得id，继续获得属性值

If(resourceId==0)

lableText=attrs.getAttributeResourceValue(null,”lableText”);

//从资源文件中获得lableText属性的值

Else lableText=getResource().getString(resourceId);

//如果以上两种方式都没有获得属性,抛出异常

If(lableText==null)

{

Throw new RuntimeException(“meiyoushuxing”);

}

//获取lableFontSize资源id

int resourceId = attrs.getAttributeResourceValue(null,”lableFontSize”,0);

//获得属性值,如果没设定,默认设置24

If(resourceId==0)//证明不是资源文件,而是可控件

lableFontSize=attrs.getAttributeIntValue(null,”lableFontSize”,24);

//从资源文件中获得属性的值

Else lableFontSize = getResource().getInteger(resourceId);

//lablePosition id

Int resourceId=attrs.getAttributeResourceValue(null,”lablePosition”,0);

If(resourceId==0)

labelPosition = attrs.getAttributeValue(null, "labelPosition");

// 从资源文件中获得labelPosition属性的值

else

labelPosition = getResources().getString(resourceId);

// 如果未设置labelPosition属性值，将该属性值设为left

if (labelPosition == null)

labelPosition = "left";

String infService = Context.LAYOUT\_INFLATER\_SERVICE;

LayoutInflater li;

// 获得LAYOUT\_INFLATER\_SERVICE服务

li = (LayoutInflater) context.getSystemService(infService);

LinearLayout linearLayout = null;

// 根据labelPosition属性的值装载不同的布局文件

if("left".equals(labelPosition))

linearLayout = (LinearLayout)li.inflate(R.layout.labeledittext\_horizontal, this);

else if("top".equals(labelPosition))

linearLayout = (LinearLayout)li.inflate(R.layout.labeledittext\_vertical, this);

else

throw new RuntimeException("labelPosition属性的值只能是left或top.");

// 下面的代码从相应的布局文件中获得了TextView对象，并根据LabelTextView的属性值设置TextView的属性

textView = (TextView) findViewById(R.id.textview);

//textView.setTextSize((float)labelFontSize);

textView.setTextSize(labelFontSize);

textView.setText(labelText);

}

}

### 布局中实现,layout

<cn.eoe.widget.lableEditText

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

lableText="第一个图标" lablePosition=”Right” lableFontSize=”55” />

2.

public class IconTextView extends TextView

{

// 命名空间的值

private final String namespace = "http://cn.eoe.icon.textview";

// 图像资源ID

private int resourceId = 0;

private Bitmap bitmap;

public IconTextView(Context context, AttributeSet attrs)

{

super(context, attrs);

resourceId = attrs.getAttributeResourceValue(namespace, "iconSrc", 0);

if (resourceId > 0)

bitmap = BitmapFactory.decodeResource(getResources(), resourceId);

}

/\*\*

\* 这个效果其实也可以通过上个方法继承LinearLayout来实现,上个是用自己的ondraw因为布局本来就是写好的，所以不用重写ondraw

\* ，而这个其实是直接画出来的。

\*/

@Override

protected void onDraw(Canvas canvas)

{

if (bitmap != null)

{

// 从原图上截取图像的区域，在本例中为整个图像

Rect src = new Rect();

// 将截取的图像复制到bitmap上的目标区域，在本例中与复制区域相同

Rect target = new Rect();

src.left = 0;

src.top = 0;

src.right = bitmap.getWidth();

src.bottom = bitmap.getHeight();

int textHeight = (int) getTextSize();

target.left = 0;

// 计算图像复制到目录区域的纵坐标。由于TextView中文本内容并不是从最顶端开始绘制的，因此，需要重新计算绘制图像的纵坐标

target.top = (int) ((getMeasuredHeight() - getTextSize()) / 2) + 1;

target.bottom = target.top + textHeight;

// 为了保证图像不变形，需要根据图像高度重新计算图像的宽度

target.right = (int) (textHeight \* (bitmap.getWidth() / (float) bitmap

.getHeight()));

// 开始绘制图像

canvas.drawBitmap(bitmap, src, target, getPaint());

// 将TextView中的文本向右移动一定的距离（在本例中移动了图像宽度加2个象素点的位置）

canvas.translate(target.right + 2, 0);

}

super.onDraw(canvas);

}

}

<?xml version="1.0" encoding="utf-8"?>

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

xmlns:mobile="http://cn.eoe.icon.textview" android:orientation="vertical"

android:layout\_width="fill\_parent" android:layout\_height="fill\_parent">

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第一个图标" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第二个图标" android:textSize="24dp" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第三个图标" android:textSize="36dp" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第四个图标" android:textSize="48dp" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第五个图标" android:textSize="36dp" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第六个图标" android:textSize="24dp" mobile:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第七个图标" mobile:iconSrc="@drawable/android" />

</LinearLayout>

### 另外一种实现方法用TypeArray.

<resources>

<attr name="iconPosition">

<enum name="left" value="0" />

<enum name="right" value="1" />

</attr>

<declare-styleable name="IconTextView">

<attr name="iconSrc" format="reference" />

<attr name="iconPosition" />

</declare-styleable>

</resources>

package cn.eoe.widget;

import android.content.Context;

import android.content.res.TypedArray;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Rect;

import android.util.AttributeSet;

import android.widget.TextView;

import cn.eoe.icon.textview.ext.R;

public class IconTextView extends TextView

{

// 图像资源ID

private int resourceId = 0;

// icon位置 0：left 1:right

private int iconPosition = 0;

private Bitmap bitmap;

public IconTextView(Context context, AttributeSet attrs)

{

super(context, attrs);

/\*\*

\* http://blog.csdn.net/lilu\_leo/article/details/7449973

\*/

TypedArray typedArray = context.obtainStyledAttributes(attrs,

R.styleable.IconTextView);

resourceId = typedArray.getResourceId(R.styleable.IconTextView\_iconSrc,

0);

if (resourceId > 0)

bitmap = BitmapFactory.decodeResource(getResources(), resourceId);

iconPosition = typedArray.getInt(R.styleable.IconTextView\_iconPosition,

0);

}

@Override

protected void onDraw(Canvas canvas)

{

if (bitmap != null)

{

// 从原图上截取图像的区域，在本例中为整个图像

Rect src = new Rect();

// 将截取的图像复制到bitmap上的目标区域，在本例中与复制区域相同

Rect target = new Rect();

src.left = 0;

src.top = 0;

src.right = bitmap.getWidth();

src.bottom = bitmap.getHeight();

int textHeight = (int) getTextSize();

int left = 0;

if (iconPosition == 1)

{

left = (int) getPaint().measureText(getText().toString()) + 2;

}

target.left = left;

// 计算图像复制到目录区域的纵坐标。由于TextView中文本内容并不是从最顶端开始绘制的，因此，需要重新计算绘制图像的纵坐标

target.top = (int) ((getMeasuredHeight() - getTextSize()) / 2) + 1;

target.bottom = target.top + textHeight;

// 为了保证图像不变形，需要根据图像高度重新计算图像的宽度

target.right = left

+ (int) (textHeight \* (bitmap.getWidth() / (float) bitmap

.getHeight()));

// 开始绘制图像

canvas.drawBitmap(bitmap, src, target, getPaint());

// 将TextView中的文本向右移动一定的距离（在本例中移动了图像宽度加2个象素点的位置）

if (iconPosition == 0)

canvas.translate(target.right + 2, 0);

}

super.onDraw(canvas);

}

}

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

xmlns:app="http://schemas.android.com/apk/res/cn.eoe.icon.textview.ext"

android:orientation="vertical" android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第一个图标" app:iconSrc="@drawable/android" app:iconPosition="left" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第二个图标" android:textSize="24sp" app:iconSrc="@drawable/android"

app:iconPosition="right" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第三个图标" android:textSize="36sp" app:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第四个图标" android:textSize="48sp" app:iconSrc="@drawable/android"

app:iconPosition="right" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第五个图标" android:textSize="36sp" app:iconSrc="@drawable/android" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第六个图标" android:textSize="24sp" app:iconSrc="@drawable/android"

app:iconPosition="right" />

<cn.eoe.widget.IconTextView

android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"

android:text="第七个图标" app:iconSrc="@drawable/android" />

</LinearLayout>