ViewPager。它是google SDk中自带的一个附加包的一个类，可以用来实现屏幕间的切换。

### 1.在主布局文件里加入

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<RelativeLayout** xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="fill\_parent"
5. tools:context="com.example.testviewpage\_1.MainActivity" **>**
7. **<android.support.v4.view.ViewPager**
8. android:id="@+id/viewpager"
9. android:layout\_width="wrap\_content"
10. android:layout\_height="wrap\_content"
11. android:layout\_gravity="center" **/>**
13. **</RelativeLayout>**

其中 <**[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/harvic880925/article/details/_blank)**.support.v4.view.ViewPager /> 是**ViewPager对应的组件，要将其放到想要滑动的位置**

### 2、新建三个layout，用于滑动切换的视图

从效果图中也可以看到，我们的三个视图都非常简单，里面没有任何的控件，大家当然可以往里添加各种控件，但这里是个DEMO，只详解原理即可，所以我这里仅仅用背景来区别不用layout布局。

**布局代码分别如下：**

layout1.xml

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. android:background="#ffffff"
6. android:orientation="vertical" **>**

9. **</LinearLayout>**

layout2.xml

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. android:background="#ffff00"
6. android:orientation="vertical" **>**

9. **</LinearLayout>**

layout3.xml

**[html]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. android:background="#ff00ff"
6. android:orientation="vertical" **>**

9. **</LinearLayout><span** style="color:#660000;"**>**
10. **</span>**

## 二、代码实战

先上整体代码，然后逐步讲解。

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **package** com.example.testviewpage\_1;
2. /\*\*
3. \* @author  harvic
4. \* @date 2014.8.9
5. \*/
6. **import** java.util.ArrayList;
7. **import** java.util.List;
8. **import** java.util.zip.Inflater;
10. **import** android.app.Activity;
11. **import** android.os.Bundle;
12. **import** android.support.v4.view.PagerAdapter;
13. **import** android.support.v4.view.ViewPager;
14. **import** android.view.LayoutInflater;
15. **import** android.view.View;
16. **import** android.view.ViewGroup;

19. **public** **class** MainActivity **extends** Activity {
21. **private** View view1, view2, view3;
22. **private** ViewPager viewPager;  //对应的viewPager
24. **private** List<View> viewList;//view数组

27. @Override
28. **protected** **void** onCreate(Bundle savedInstanceState) {
29. **super**.onCreate(savedInstanceState);
30. setContentView(R.layout.activity\_main);
32. viewPager = (ViewPager) findViewById(R.id.viewpager);
33. LayoutInflater inflater=getLayoutInflater();
34. view1 = inflater.inflate(R.layout.layout1, **null**);
35. view2 = inflater.inflate(R.layout.layout2,**null**);
36. view3 = inflater.inflate(R.layout.layout3, **null**);
38. viewList = **new** ArrayList<View>();// 将要分页显示的View装入数组中
39. viewList.add(view1);
40. viewList.add(view2);
41. viewList.add(view3);

44. PagerAdapter pagerAdapter = **new** PagerAdapter() {
46. @Override
47. **public** **boolean** isViewFromObject(View arg0, Object arg1) {
48. // TODO Auto-generated method stub
49. **return** arg0 == arg1;
50. }
52. @Override
53. **public** **int** getCount() {
54. // TODO Auto-generated method stub
55. **return** viewList.size();
56. }
58. @Override
59. **public** **void** destroyItem(ViewGroup container, **int** position,
60. Object object) {
61. // TODO Auto-generated method stub
62. container.removeView(viewList.get(position));
63. }
65. @Override
66. **public** Object instantiateItem(ViewGroup container, **int** position) {
67. // TODO Auto-generated method stub
68. container.addView(viewList.get(position));

71. **return** viewList.get(position);
72. }
73. };

76. viewPager.setAdapter(pagerAdapter);
78. }

81. }

代码量很小，全部放在了OnCreate()函数中。

### 1、先看声明的变量的意义：

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. **private** View view1, view2, view3;
2. **private** List<View> viewList;//view数组
3. **private** ViewPager viewPager;  //对应的viewPager

首先viewPager对应 <**[android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/harvic880925/article/details/_blank)**.support.v4.view.ViewPager/>控件。

view1,view2 ,view3对应我们的三个layout，即layout1.xml,layout2.xml,layout3.xml

viewList是一个View数组，盛装上面的三个VIEW

### 2、接下来是他们的初始化过程：

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. viewPager = (ViewPager) findViewById(R.id.viewpager);
2. LayoutInflater inflater=getLayoutInflater();
3. view1 = inflater.inflate(R.layout.layout1, **null**);
4. view2 = inflater.inflate(R.layout.layout2,**null**);
5. view3 = inflater.inflate(R.layout.layout3, **null**);
7. viewList = **new** ArrayList<View>();// 将要分页显示的View装入数组中
8. viewList.add(view1);
9. viewList.add(view2);
10. viewList.add(view3);

初始化过程难度不大，就是将资源与变量联系起来布局，最后将实例化的view1,view2,view3添加到viewList中

### 3、PageAdapter——PageView的适配器

适配器这个东东想必大家都不莫生，在ListView中也有适配器，listView通过重写GetView（）函数来获取当前要加载的Item。而PageAdapter不太相同，毕竟PageAdapter是单个VIew的合集。

**PageAdapter 必须重写的四个函数：**

* boolean isViewFromObject(View arg0, Object arg1)
* int getCount()
* void destroyItem(ViewGroup container, int position,Object object)
* Object instantiateItem(ViewGroup container, int position)

**先看看各个函数，我们上面都做了什么吧：**

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. @Override
2. **public** **int** getCount() {
3. // TODO Auto-generated method stub
4. **return** viewList.size();
5. }

getCount():返回要滑动的VIew的个数

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. @Override
2. **public** **void** destroyItem(ViewGroup container, **int** position,
3. Object object) {
4. // TODO Auto-generated method stub
5. container.removeView(viewList.get(position));
6. }

destroyItem（）：从当前container中删除指定位置（position）的View;

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. @Override
2. **public** Object instantiateItem(ViewGroup container, **int** position) {
3. // TODO Auto-generated method stub
4. container.addView(viewList.get(position));

7. **return** viewList.get(position);
8. }
9. };

instantiateItem()：做了两件事，第一：将当前视图添加到container中，第二：返回当前View

**[java]** [view plain](http://blog.csdn.net/harvic880925/article/details/38453725" \o "view plain" \t "http://blog.csdn.net/harvic880925/article/details/_blank) [copy](http://blog.csdn.net/harvic880925/article/details/38453725" \o "copy" \t "http://blog.csdn.net/harvic880925/article/details/_blank)

1. @Override
2. **public** **boolean** isViewFromObject(View arg0, Object arg1) {
3. // TODO Auto-generated method stub
4. **return** arg0 == arg1;
5. }

isViewFromObject():对于这个函数就先不做讲解，大家目前先知道它要这样重写就行了，后面我们会对它进行改写。