

2.6.3 ViewPager的简单使用

分类 [Android 基础入门教程](#)

本节引言：

本节带来的是Android 3.0后引入的一个UI控件——ViewPager(视图滑动切换工具)，实在想不到 如何来称呼这个控件，他的大概功能：通过手势滑动可以完成View的切换，一般是用来做APP 的引导页或者实现图片轮播，因为是3.0后引入的，如果想在低版本下使用，就需要引入v4 兼容包哦~，我们也可以看到，ViewPager在：
android.support.v4.view.ViewPager目录下~ 下面我们就来学习一下这个控件的基本用法~ 官方API文档：[ViewPager](#)

1.ViewPager的简单介绍

ViewPager就是一个简单的页面切换组件，我们可以往里面填充多个View，然后我们可以左 右滑动，从而切换不同的View，我们可以通过setPageTransformer()方法为我们的ViewPager 设置切换时的动画效果，当然，动画我们还没学到，所以我们把为ViewPager设置动画 放到下一章绘图与动画来讲解！和前面学的ListView，GridView一样，我们也需要一个Adapter (适配器)将我们的View和ViewPager进行绑定，而ViewPager则有一个特定的Adapter——
PagerAdapter！另外，Google官方是建议我们使用Fragment来填充ViewPager的，这样 可以更加方便的生成每个Page，以及管理每个Page的生命周期！给我们提供了两个Fragment 专用的Adapter：**FragmentPagerAdapter**和**FragmentStatePagerAdapter** 我们简要的分析下这两个Adapter的区别：

FragmentPagerAdapter：和PagerAdapter一样，只会缓存当前的Fragment以及左边一个，右边 一个，即总共会缓存3个Fragment而已，假如有1，2，3，4四个页面：

处于1页面：缓存1，2

处于2页面：缓存1，2，3

处于3页面：销毁1页面，缓存2，3，4

处于4页面：销毁2页面，缓存3，4

更多页面的情况，依次类推~

FragmentStatePagerAdapter：当Fragment对用户不 见得时，整个Fragment会被销毁， 只会保存Fragment的状态！而在页面需要重新显示的时候，会生成新的页面！

综上，FragmentPagerAdapter适合固定的页面较少的场合；而

**Android 基础入门教程(Q群号：
153836263)**

- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
- 1.1 背景相关与系统架构分析
- 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
- 1.3 SDK更新不了问题解决
- 1.4 Genymotion模拟器安装
- 1.5.1 Git使用教程之本地仓...
- 1.5.2 Git之使用GitHub搭建...
- 1.6 .9(九妹)图片怎么玩
- 1.7 界面原型设计
- 1.8 工程相关解析(各种文件...
- 1.9 Android程序签名打包
- 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
- 2.3.1 TextView(文本框)详解
- 2.3.2 EditText(输入框)详解
- 2.3.3 Button(按钮)与ImageB...
- 2.3.4 ImageView(图像视图)
- 2.3.5.RadioButton(单选按钮...
- 2.3.6 开关按钮ToggleButton...
- 2.3.7 ProgressBar(进度条)
- 2.3.8 SeekBar(拖动条)
- 2.3.9 RatingBar(星级评分条)
- 2.4.1 ScrollView(滚动条)

FragmentManager则适合于页面较多或者页面内容非常复杂(需占用大量内存)的情况！

2. PagerAdapter的使用

我们先来介绍最普通的PagerAdapter，如果想使用这个PagerAdapter需要重写下面的四个方法：当然，这只是官方建议，实际上我们只需重写getCount()和isViewFromObject()就可以了~

getCount():获得viewpager中有多少个view

destroyItem():移除一个给定位置的页面。适配器有责任从容器中删除这个视图。这是为了确保在finishUpdate(viewGroup)返回时视图能够被移除。

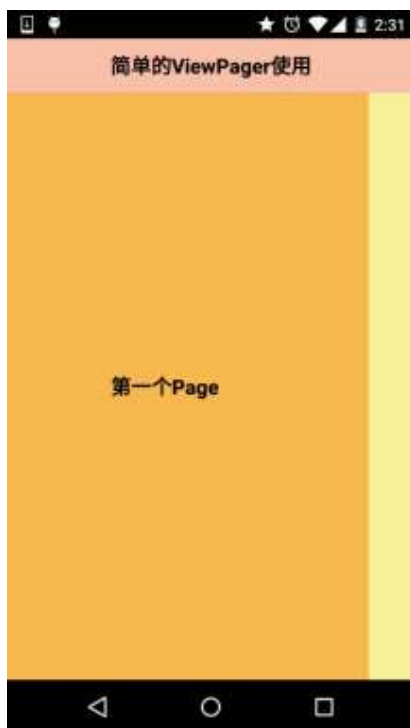
而另外两个方法则是涉及到一个key的东东：

instantiateItem(): ①将给定位置的view添加到ViewGroup(容器)中,创建并显示出来 ②返回一个代表新增页面的Object(key),通常都是直接返回view本身就可以了,当然你也可以自定义自己的key,但是key和每个view要一一对应的关系

isViewFromObject(): 判断instantiateItem(ViewGroup, int)函数所返回来的Key与一个页面视图是否是代表的同一个视图(即它俩是否是对应的，对应的表示同一个View),通常我们直接写 return view == object!

使用示例1：最简单用法

运行效果图：



关键部分代码：

好的，代码写起来也是非常简单的：首先是每个View的布局，一式三份，另外两个View一样：

view_one.xml：

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

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定...

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextVie...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框)...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTo...

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(Co...

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFBA55"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="第一个Page"
        android:textColor="#000000"
        android:textSize="18sp"
        android:textStyle="bold" />

</LinearLayout>
```



反馈

然后编写一个自定义的PagerAdapter :

MyPagerAdapter.java :

```
public class MyPagerAdapter extends PagerAdapter {
    private ArrayList<View> viewLists;

    public MyPagerAdapter() {
    }

    public MyPagerAdapter(ArrayList<View> viewLists) {
        super();
        this.viewLists = viewLists;
    }

    @Override
    public int getCount() {
        return viewLists.size();
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }

    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        container.addView(viewLists.get(position));
        return viewLists.get(position);
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        container.removeView(viewLists.get(position));
    }
}
```

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲一...

5.2.2 Fragment实例精讲一...

5.2.3 Fragment实例精讲一...

5.2.4 Fragment实例精讲一...

5.2.5 Fragment实例精讲一...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

接着到Activity了，和以前学的ListView非常类似：

OneActivity.java：

```
public class OneActivity extends AppCompatActivity{

    private ViewPager vpager_one;
    private ArrayList<View> aList;
    private MyPagerAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
        vpager_one = (ViewPager) findViewById(R.id.vpager_one);

        aList = new ArrayList<View>();
        LayoutInflater li = getLayoutInflater();
        aList.add(li.inflate(R.layout.view_one,null,false));
        aList.add(li.inflate(R.layout.view_two,null,false));
        aList.add(li.inflate(R.layout.view_three,null,false));
        mAdapter = new MyPagerAdapter(aList);
        vpager_one.setAdapter(mAdapter);
    }
}
```

好的，关键代码就上述部分，非常容易理解~

使用示例2：标题栏——PagerTitleStrip与PagerTabStrip

就是跟随着ViewPager滑动而滑动的标题咯，这两个是官方提供的，一个是普通文字，一个是带有下列线，以及可以点击文字可切换页面，下面我们来写个简单的例子~

运行效果图：

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...



关键代码实现：

这里两者的区别仅仅是布局不一样而已，其他的都一样：

PagerTitleStrip所在Activitiy的布局：**activity_two.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="#CCFF99"
        android:gravity="center"
        android:text="PagerTitleStrip效果演示"
        android:textColor="#000000"
        android:textSize="18sp" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager_two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <android.support.v4.view.PagerTitleStrip
            android:id="@+id/pagertitle"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:layout_gravity="top"
            android:textColor="#FFFFFF" />
    </android.support.v4.view.ViewPager>

</LinearLayout>
```

而PagerTabStrip所在的布局：

activity_three.xml：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:background="#C0C080"
        android:gravity="center"
        android:text="PagerTabStrip效果演示"
        android:textSize="18sp" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager_three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <android.support.v4.view.PagerTabStrip
            android:id="@+id/pagertitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top" />

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

接下来的两者都一样了，我们先来编写一个自定义的PagerAdapter，除了前面重写的四个方法外，我们需要另外重写一个方法：**getPageTitle()**，这个设置标题的~代码如下：

MyPagerAdapter2.java：

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class MyPagerAdapter2 extends PagerAdapter {
    private ArrayList<View> viewLists;
    private ArrayList<String> titleLists;

    public MyPagerAdapter2() {}
    public MyPagerAdapter2(ArrayList<View> viewLists, ArrayList<
String> titleLists)
    {
        this.viewLists = viewLists;
        this.titleLists = titleLists;
    }
}
```

```

@Override
public int getCount() {
    return viewLists.size();
}

@Override
public boolean isViewFromObject(View view, Object object) {
    return view == object;
}

@Override
public Object instantiateItem(ViewGroup container, int position) {
    container.addView(viewLists.get(position));
    return viewLists.get(position);
}

@Override
public void destroyItem(ViewGroup container, int position,
Object object) {
    container.removeView(viewLists.get(position));
}

@Override
public CharSequence getPageTitle(int position) {
    return titleLists.get(position);
}
}

```

最后是Activity部分，两个都是一样的：

TwoActivity.java :

```

/**
 * Created by Jay on 2015/10/8 0008.
 */
public class TwoActivity extends AppCompatActivity {

    private ViewPager vpager_two;
    private ArrayList<View> aList;
    private ArrayList<String> sList;
    private MyPagerAdapter2 mAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);
        vpager_two = (ViewPager) findViewById(R.id.vpager_two);
        aList = new ArrayList<View>();
        LayoutInflater li = getLayoutInflater();
        aList.add(li.inflate(R.layout.view_one, null, false));
        aList.add(li.inflate(R.layout.view_two, null, false));
        aList.add(li.inflate(R.layout.view_three, null, false));

        sList = new ArrayList<String>();
    }
}

```

```
sList.add("橘黄");
sList.add("淡黄");
sList.add("浅棕");
mAdapter = new MyPagerAdapter2(aList,sList);
vpager_two.setAdapter(mAdapter);
}
}
```

好了，非常简单，有疑问的话，自己下demo看看就懂了~

使用示例3：ViewPager实现TabHost的效果：

当然，示例2很多时候，只是中看不中用，实际开发中我们可能需要自行定制这个标题栏，下面我们就来写个简单的例子来实现TabHost的效果，如果你不知道TabHost是什么鬼的话，那么，请看效果图！

运行效果图：



实现逻辑解析：

下面我们来讲解下实现上述效果的逻辑，然后贴代码：

首先是布局：顶部一个LinearLayout，包着三个TextView，weight属性都为1，然后下面跟着一个滑块的ImageView，我们设置宽度为match_parent；最底下是我们的ViewPager，这里可能有两个属性你并不认识，一个是：flipInterval：这个是指定View动画间的时间间隔的！

而persistentDrawingCache：则是设置控件的绘制缓存策略，可选值有四个：

none：不在内存中保存绘图缓存；

animation:只保存动画绘图缓存；

scrolling：只保存滚动效果绘图缓存；

all : 所有的绘图缓存都应该保存在内存中 ;

可以同时用2个 , animation|scrolling这样~

布局代码 : **activity_four.xml** :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="48dp"
        android:background="#FFFFFF">

        <TextView
            android:id="@+id/tv_one"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="橘黄"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/tv_two"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="淡黄"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/tv_three"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="浅棕"
            android:textColor="#000000" />
    </LinearLayout>

    <ImageView
        android:id="@+id/img_cursor"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:scaleType="matrix"
        android:src="@mipmap/line" />
```

```
<android.support.v4.view.ViewPager
    android:id="@+id/vpager_four"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_gravity="center"
    android:layout_weight="1.0"
    android:flipInterval="30"
    android:persistentDrawingCache="animation" />
```

```
</LinearLayout>
```

接着到我们的Activity了，我们来捋下思路：

Step 1：我们需要让我们的移动块在第一个文字下居中，那这里就要算一下偏移量：先获得图片宽度pw，然后获取屏幕宽度sw，计算方法很简单：

offset(偏移量) = ((sw / 3) - pw) / 2 // 屏幕宽/3 - 图片宽度，然后再除以2，左右嘛！

然后我们调用setImageMatrix设置滑块当前的位置：

同时我们也把切换一页和两页，滑块的移动距离也算出来，很简单：

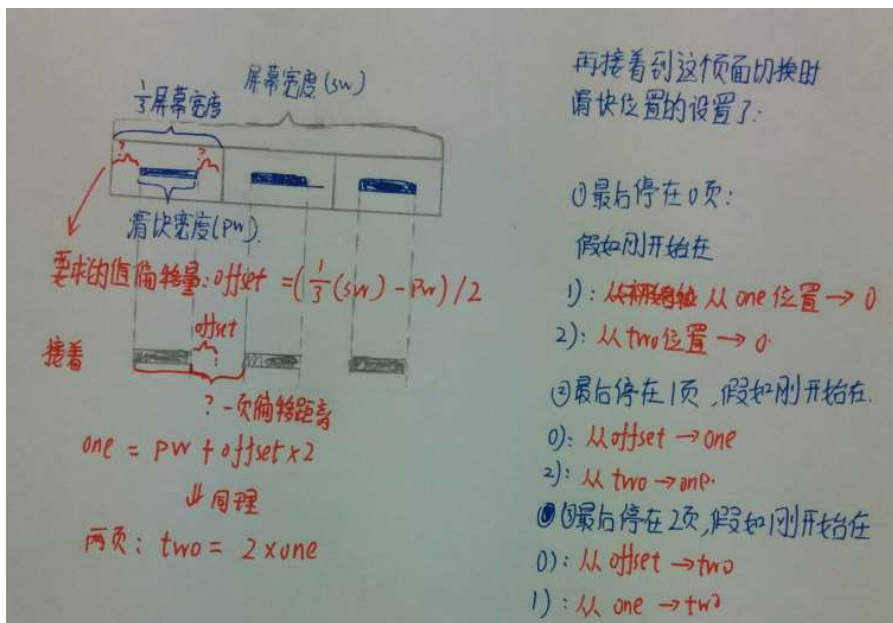
one = offset * 2 + pw;

two = one * 2;

Step 2：当我们滑动页面时，我们的滑块要进行移动，我们要为ViewPager添加一个 OnPageChangeListener事件，我们需要对滑动后的页面来做一个判断，同时记录滑动前处于哪个页面，下面自己画了个图，可能更容易理解吧！



PS:太久没写字，字很丑，能看清就好，字丑人美，哈哈~



嗯，如果还是不能理解的话，自己动手画画图就知道了，下面上代码：

FourActivitiy.java :

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class FourActivity extends AppCompatActivity implements
    View.OnClickListener,
    ViewPager.OnPageChangeListener {

    private ViewPager vpager_four;
    private ImageView img_cursor;
    private TextView tv_one;
    private TextView tv_two;
    private TextView tv_three;

    private ArrayList<View> listViews;
    private int offset = 0; //移动条图片的偏移量
    private int currIndex = 0; //当前页面的编号
    private int bmpWidth; // 移动条图片的长度
    private int one = 0; //移动条滑动一页的距离
    private int two = 0; //滑动条移动两页的距离

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_four);
        initView();
    }

    private void initView() {
        vpager_four = (ViewPager) findViewById(R.id.vpager_four
    );

        tv_one = (TextView) findViewById(R.id.tv_one);
        tv_two = (TextView) findViewById(R.id.tv_two);
        tv_three = (TextView) findViewById(R.id.tv_three);
        img_cursor = (ImageView) findViewById(R.id.img_cursor);

        //下划线动画的相关设置:
        bmpWidth = BitmapFactory.decodeResource(getResources(),
R.mipmap.line).getWidth(); // 获取图片宽度
        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        int screenW = dm.widthPixels; // 获取分辨率宽度
        offset = (screenW / 3 - bmpWidth) / 2; // 计算偏移量
        Matrix matrix = new Matrix();
        matrix.postTranslate(offset, 0);
        img_cursor.setImageMatrix(matrix); // 设置动画初始位置
        //移动的距离
        one = offset * 2 + bmpWidth; // 移动一页的偏移量,比如1->2,
或者2->3
        two = one * 2; // 移动两页的偏移量,比如1直接跳3

        //往ViewPager填充View, 同时设置点击事件与页面切换事件
        listViews = new ArrayList<View>();
        LayoutInflater mInflater = getLayoutInflater();
```

```

        listView.add(mInflater.inflate(R.layout.view_one, null
, false));
        listView.add(mInflater.inflate(R.layout.view_two, null
, false));
        listView.add(mInflater.inflate(R.layout.view_three, nu
ll, false));
        vpager_four.setAdapter(new MyPagerAdapter(listViews));
        vpager_four.setCurrentItem(0);           //设置ViewPager
当前页, 从0开始算

        tv_one.setOnClickListener(this);
        tv_two.setOnClickListener(this);
        tv_three.setOnClickListener(this);

        vpager_four.addOnPageChangeListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.tv_one:
                vpager_four.setCurrentItem(0);
                break;
            case R.id.tv_two:
                vpager_four.setCurrentItem(1);
                break;
            case R.id.tv_three:
                vpager_four.setCurrentItem(2);
                break;
        }
    }

    @Override
    public void onPageSelected(int index) {
        Animation animation = null;
        switch (index) {
            case 0:
                if (currIndex == 1) {
                    animation = new TranslateAnimation(one, 0,
0, 0);
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, 0,
0, 0);
                }
                break;
            case 1:
                if (currIndex == 0) {
                    animation = new TranslateAnimation(offset,
one, 0, 0);
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, one
, 0, 0);
                }
                break;
            case 2:

```