[Android ViewPager使用方法小结](https://www.cnblogs.com/jzyhywxz/p/6914307.html)

android-support-v4.jar 是谷歌提供给我们的一个兼容低版本安卓设备的软件包，里面包囊了只有在 Android 3.0 以上可用的API。而 ViewPager 就是其中之一。利用它，我们可以做很多事情，从最简单的导航，到页面菜单等等。

**准备**

在使用ViewPager之前，需要在 build.gradle 中加入如下语句：

compile 'com.android.support:support-v4:25.3.0'

compile 'com.github.hackware1993:MagicIndicator:1.5.0'

其中第一行是 android-support-v4 软件包，ViewPager 就在里面；第二行是一个开源的导航器，通过它，我们可以实现各种导航效果。

**ViewPager -- 基本用法**

ViewPager 的基本用法可以分为以下步骤：

1. 在 layout 文件中定义一个 ViewPager 组件；
2. 在 Activity（或Fragment等）中取得 ViewPager 的引用；
3. 为 ViewPager 设置适配器；
4. [为 ViewPager 设置滑动特效]；
5. [为 ViewPager 设置监听器]。

其中最后两步是可选的，但如果没有最后两步，那我们的 ViewPager 就仅仅是一个普普通通的 ViewPager 了。  
  
下面我们来写一个最简单的例子，这里事先在drawable文件夹中放入了一些图片，并且这些图片都以 `page` 开头：

[复制代码](javascript:void(0);)

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

<RelativeLayout 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"

tools:context="com.zzw.activity.PageActivity">

<android.support.v4.view.ViewPager

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/view\_pager"

android:background="@android:color/black" />

</RelativeLayout>

[复制代码](javascript:void(0);)

我们在布局文件里定义了一个 ViewPager 组件，然后需要在 Activity 中对它进行设置：

[复制代码](javascript:void(0);)

pager=(ViewPager) findViewById(R.id.view\_pager);

PagerAdapter adapter=new ViewAdapter(pages);

pager.setAdapter(adapter);

class ViewAdapter extends PagerAdapter {

private List<View> datas;

public ViewAdapter(List<View> list) {

datas=list;

}

@Override

public int getCount() {

return datas.size();

}

@Override

public boolean isViewFromObject(View view, Object object) {

return view==object;

}

@Override

public Object instantiateItem(ViewGroup container, int position) {

View view=datas.get(position);

container.addView(view);

return view;

}

@Override

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

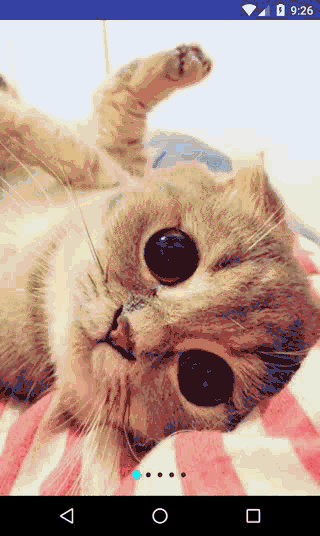
container.removeView(datas.get(position));

}

}

[复制代码](javascript:void(0);)

嗯，现在我们就完成了一个“图片浏览器”，效果和下图类似：

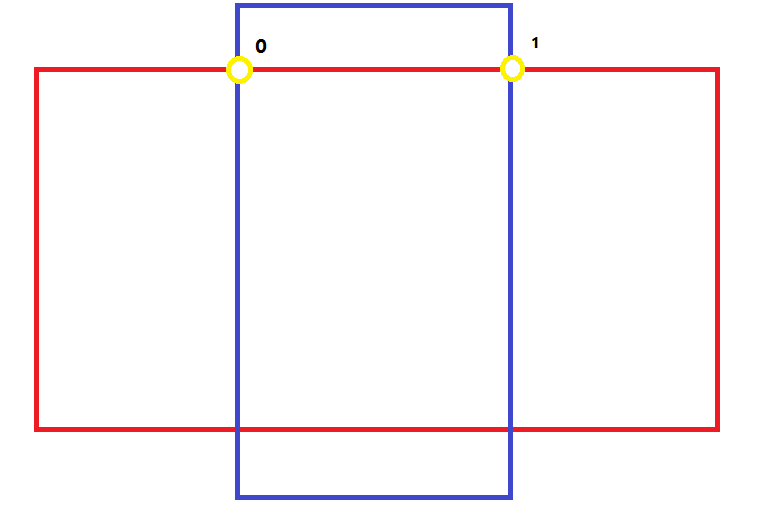


*PS：你可能注意到，我们的效果图底部有一个导航器，这是因为这个效果图是加上导航器之后的效果（实在懒得回头再改一遍了）。*  
*那你可能会问，这个导航器怎么实现的，为啥上面的代码里没有？别急，因为这个开源的导航器实在太牛B了，所以我想专门用一节来介绍，如果你等不及的话，也可以先去看导航器的介绍。*

好了，现在我们已经基本熟悉了 ViewPager 的基本使用方法，但是！ViewPager 的效果还远远不止于此，我们接着往下面看。

**ViewPager -- 浮现特效**

嗯，现在我们要在上面的“图片浏览器”基础上加点特效了（要不然用户肯定审美疲劳了）。试想，当我们滑动图片时，如果下一张图片能从后面“浮现”出来，是不是很棒！好了，那我们就来尝试添加这一个特效。  
  
但是，要从哪里入手呢？哈哈，其实Google早就为我们提供了相应的API，就是 android.support.v4.view.ViewPager.PageTransformer 这个接口，任何实现了这个接口的类都可以为 ViewPager 提供一种特效哦。  
  
我们发现，这个接口里只有一个方法 public void transformPage(View page, float position) ，看看它的两个参数，`page`表示 ViewPager 中的一页，`position`表示`page`当前的位置，[-1, 0)表示屏幕左边的`page`（部分可见），[0, 0]表示屏幕上的`page`（完全可见），(0, 1]表示屏幕右边的`page`（部分可见），具体看下图：

  
当`page`向左边滑动时，`position`从0向-1变化，当`position==-1`时完全不可见；当`page`向右滑动时，`position`从0向1变化，当`position==1`时完全不可见。  
  
如果你想更加深入的了解`position`的规律（可能是我没表达清楚），可以参考鸿洋大神的文章[《Android 实现个性的ViewPager切换动画 实战PageTransformer（兼容Android3.0以下）》](http://blog.csdn.net/lmj623565791/article/details/40411921/)。  
  
好了，经过上面的说明，我们发现只需要处理[-1, 1]范围的`position`就行了：

[复制代码](javascript:void(0);)

public class ScalePageTransformer implements ViewPager.PageTransformer {

private static final float MIN\_SCALE=0.75f;

@Override

public void transformPage(View page, float position) {

//Log.d("TAG", "<"+page.hashCode()+", "+position+">");

// out of left screen

if(position<-1.0f) {

page.setScaleX(MIN\_SCALE);

page.setScaleY(MIN\_SCALE);

}

// slide left

else if(position<=0.0f) {

page.setAlpha(1.0f);

page.setTranslationX(0.0f);

page.setScaleX(1.0f);

page.setScaleY(1.0f);

}

// slide right

else if(position<=1.0f) {

page.setAlpha(1.0f-position);

page.setTranslationX(-page.getWidth()\*position);

float scale=MIN\_SCALE+(1.0f-MIN\_SCALE)\*(1.0f-position);

page.setScaleX(scale);

page.setScaleY(scale);

}

// out of right screen

else {

page.setScaleX(MIN\_SCALE);

page.setScaleY(MIN\_SCALE);

}

}

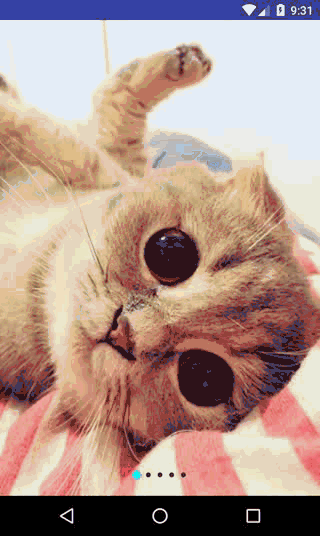
}

[复制代码](javascript:void(0);)

然后还要为 ViewPager 添加这个特效：

pager.setPageTransformer(true, new ScalePageTransformer());

现在的效果如下：

  
哈哈，我们的“图片浏览器”是不是好看了一大截。但是，别急着走，我们还可以实现更多的特效。

**ViewPager -- 旋转特效**

有了上面的基础，下面我们再来做一个旋转特效吧：

[复制代码](javascript:void(0);)

public class RotatePageTransformer implements ViewPager.PageTransformer {

private static final float MAX\_ROTATION=20.0f;

@Override

public void transformPage(View page, float position) {

if(position<-1)

rotate(page, -MAX\_ROTATION);

else if(position<=1)

rotate(page, MAX\_ROTATION\*position);

else

rotate(page, MAX\_ROTATION);

}

private void rotate(View view, float rotation) {

view.setPivotX(view.getWidth()\*0.5f);

view.setPivotY(view.getHeight());

view.setRotation(rotation);

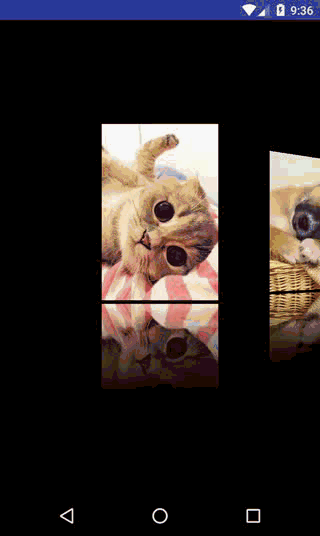
}

}

[复制代码](javascript:void(0);)

这个更简单，只需要根据`position`来决定旋转的角度即可。效果如下：  


**ViewPager -- 3D画廊**

下面我们再实现一个特效，不过这个特效稍微复杂，但是效果真的挺炫！我们先来看下效果：  
  
是不是就像自己身处3D画廊一样。好了，下面我们就来实现这个特效。  
  
首先还是 PageTransformer：

[复制代码](javascript:void(0);)

public class GalleryPageTransformer implements ViewPager.PageTransformer {

private static final float MAX\_ROTATION=20.0f;

private static final float MIN\_SCALE=0.75f;

private static final float MAX\_TRANSLATE=20.0f;

@Override

public void transformPage(View page, float position) {

if(position<-1) {

page.setTranslationX(MAX\_TRANSLATE);

page.setScaleX(MIN\_SCALE);

page.setScaleY(MIN\_SCALE);

page.setRotationY(-MAX\_ROTATION);

}

else if(position<=0) {

page.setTranslationX(-MAX\_TRANSLATE\*position);

float scale=MIN\_SCALE+(1-MIN\_SCALE)\*(1.0f+position);

page.setScaleX(scale);

page.setScaleY(scale);

page.setRotationY(MAX\_ROTATION\*position);

}

else if(position<=1) {

page.setTranslationX(-MAX\_TRANSLATE\*position);

float scale=MIN\_SCALE+(1-MIN\_SCALE)\*(1.0f-position);

page.setScaleX(scale);

page.setScaleY(scale);

page.setRotationY(MAX\_ROTATION\*position);

}

else {

page.setTranslationX(-MAX\_TRANSLATE);

page.setScaleX(MIN\_SCALE);

page.setScaleY(MIN\_SCALE);

page.setRotationY(MAX\_ROTATION);

}

}

}

[复制代码](javascript:void(0);)

这里的“3D”特效主要是靠 View.setRotationY(float rotation) 方法实现的，此方法的作用是让 View 绕着Y轴旋转一定角度（关于Android的三维坐标系请大家自己查询）。  
  
看到这，可能你会问，倒影去哪了？别急，我们并不是在 PageTransformer 里实现倒影的，而是一开始就有的。没错，我们在为 ViewPager 设置适配器时，就直接把有倒影的图片封装到适配器中了。  
  
下面的方法负责生成有倒影的图片：

[复制代码](javascript:void(0);)

public static Bitmap getReverseBitmapById(Context context, int resId, float percent) {

// get the source bitmap

Bitmap srcBitmap=BitmapFactory.decodeResource(context.getResources(), resId);

// get the tow third segment of the reverse bitmap

Matrix matrix=new Matrix();

matrix.setScale(1, -1);

Bitmap rvsBitmap=Bitmap.createBitmap(srcBitmap, 0, (int) (srcBitmap.getHeight()\*(1-percent)),

srcBitmap.getWidth(), (int) (srcBitmap.getHeight()\*percent), matrix, false);

// combine the source bitmap and the reverse bitmap

Bitmap comBitmap=Bitmap.createBitmap(srcBitmap.getWidth(),

srcBitmap.getHeight()+rvsBitmap.getHeight()+20, srcBitmap.getConfig());

Canvas gCanvas=new Canvas(comBitmap);

gCanvas.drawBitmap(srcBitmap, 0, 0, null);

gCanvas.drawBitmap(rvsBitmap, 0, srcBitmap.getHeight()+20, null);

Paint paint=new Paint();

LinearGradient shader=new LinearGradient(0, srcBitmap.getHeight()+20, 0, comBitmap.getHeight(),

Color.BLACK, Color.TRANSPARENT, Shader.TileMode.CLAMP);

paint.setShader(shader);

paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST\_IN));

gCanvas.drawRect(0, srcBitmap.getHeight()+20, srcBitmap.getWidth(), comBitmap.getHeight(), paint);

return comBitmap;

}

[复制代码](javascript:void(0);)

其中的`percent`参数指定倒影占原图的比例。  
  
然后对原图进行处理：

[复制代码](javascript:void(0);)

private List<View> getPages() {

List<View> pages=new ArrayList<>();

Field[] fields=R.drawable.class.getDeclaredFields();

try {

for (Field field : fields) {

if (field.getName().startsWith("page")) {

ImageView view = new ImageView(this);

view.setImageBitmap(ImageUtils.getReverseBitmapById(this, field.getInt(null), 0.5f));

pages.add(view);

}

}

} catch (IllegalAccessException e) {

e.printStackTrace();

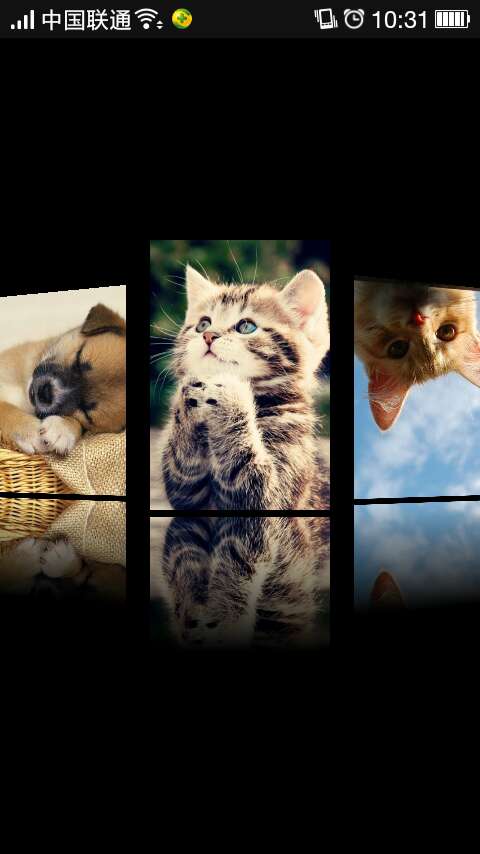
}

return pages;

}

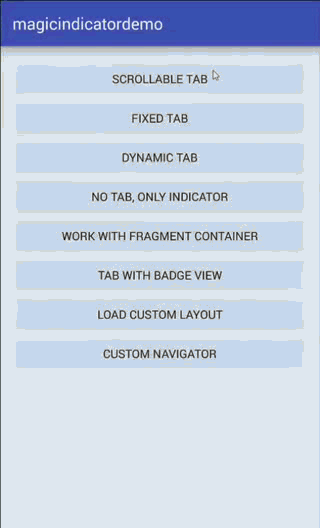
[复制代码](javascript:void(0);)

现在我们就有了一个绚丽的“3D画廊”了：



**MagicIndicator -- MAGIC！！！**

到此为止，对 ViewPager 的介绍也告一段落了。但是，你有没有发现，当我们滑动图片时，不知道当前是第几张图（我们的图片还很少，如果多的话就会有这个问题）！！！  
  
为了解决这个问题，我们还需要一个指示器/导航器来指定当前的位置。但是，一想到要从头实现一个指示器就感到头大（当然一个简单的指示器还是很容易的）。幸好，还有 MagicIndicator 来帮你。  
  
MagicIndicator，“人”如其名，它的特点就是“MAGIC”！！！不多说，先来看下它的效果图：

  
看到这效果我直接给跪了！关于 MagicIndicator 的详细介绍可以看这几篇文章：

1. [系列之一 —— 使用MagicIndicator打造千变万化的ViewPager指示器》](http://www.jianshu.com/p/2865812fed41%E3%80%8AMagicIndicator)
2. [系列之二 —— MagicIndicator使用指南》](http://www.jianshu.com/p/e9a832ed6c4c%E3%80%8AMagicIndicator)
3. [《MagicIndicator系列之三 —— MagicIndicator原理浅析及扩展MagicIndicator的4种方式》](http://www.jianshu.com/p/f3022211821c)

*MagicIndicator 的GitHub地址是：*[*https://github.com/hackware1993/MagicIndicator*](https://github.com/hackware1993/MagicIndicator)*，关于它的具体用法也在上面，这里我们就不多说了。*

***ViewPager与Fragment***

*ViewPager 经常与 Fragment 结合来实现各种页面切换，比如我们开头提到的页面菜单。由于在 ViewPager 中使用 Fragment 和上面介绍的方式稍有不同，因此我们在这里单独介绍一下。  
  
仔细想想，我们之前的 ViewPager 的页面都是 View，而现在要换成 Fragment，是不是需要替换适配器？没错，的确如此，不过我们没必要从头写一个适配器，Google为我们提供了两个抽象类：FragmentPagerAdapter 和 FragmentStatePagerAdapter。  
  
不论是继承 FragmentPagerAdapter 还是 FragmentStatePagerAdapter，都需要实现构造器、 int getCount() 和 Fragment getItem(int position) 方法。下面是我们的例子：*

[复制代码](javascript:void(0);)

public class FragmentAdapter extends FragmentPagerAdapter {

private List<Fragment> datas;

public FragmentAdapter(FragmentManager fm, List<Fragment> list) {

super(fm);

datas=list;

}

@Override

public int getCount() {

return datas.size();

}

@Override

public Fragment getItem(int position) {

return datas.get(position);

}

}

[复制代码](javascript:void(0);)

*FragmentPagerAdapter 中的每一个 Fragment 都将保存在内存之中，因此适用于那些相对静态、数量较少的情况；如果需要处理有很多页，并且数据动态性较大、占用内存较多的情况，应该使用FragmentStatePagerAdapter。  
  
FragmentStatePagerAdapter 和 FragmentPagerAdapter 一样，继承自 PagerAdapter。但和 FragmentPagerAdapter 不同的是，正如其类名中的“State”所表明的含义一样，它只保留当前页面。当页面离开视线后，就会被回收，释放其资源；而在页面需要显示时，将生成新的页面。这样的好处就是当拥有大量的页面时，不会在内存中占用大量的内存。  
  
有了 Fragment 的适配器后，还需要为 ViewPager 设置适配器：*

FragmentAdapter adapter=new FragmentAdapter(getSupportFragmentManager(), frags);

pager.setAdapter(adapter);

*最后的效果如下：*

**

***源代码***

上述所有源代码都已上传到GitHub：  
<https://github.com/jzyhywxz/ViewPager>

**标签:**[Android](https://www.cnblogs.com/jzyhywxz/tag/Android/)