ViewPager 一、二、三、四、五

PageAdapter——PageView的适配器

PageAdapter必须重写四个函数

```
* 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，都做了什么

```
@Override
public int getCount() {
    // TODO Auto-generated method stub
    return viewList.size();
}
```

getCount():返回要滑动的VIew的个数

```
@Override
public void destroyItem(ViewGroup container, int position,
        Object object) {
    // TODO Auto-generated method stub
    container.removeView(viewList.get(position));
}
```

destroyItem（）：从当前container中删除指定位置（position）的View; 这是为了确保在finishUpdate(viewGroup)返回时视图能够被移除。

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


        return viewList.get(position);
    }
};
```

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

```
@Override
public boolean isViewFromObject(View arg0, Object arg1) {
    // TODO Auto-generated method stub
    return arg0 == arg1;
}
```

该函数用来判断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) 该函数用来判断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) 。

PagerTitleStrip

官方 ： PagerTabStrip是ViewPager的一个关于当前页面、上一个页面和下一个页面的一个非交互的指示器。它经常作为ViewPager控件的一个子控件被被添加在XML布局文件中。在你的布局文件中，将它作为子控件添加在ViewPager中。而且要将它的 android:layout_gravity 属性设置为TOP或BOTTOM来将它显示在ViewPager的顶部或底部。每个页面的标题是通过适配器的getPageTitle(int)函数提供给ViewPager的。

```
1、首先，文中提到：在你的布局文件中，将它作为子控件添加在ViewPager中。

2、第二，标题的获取，是重写适配器的getPageTitle(int)函数来获取的。
```

PagerTabStrip

```
1. PagerTabStrip在当前页面下，会有一个下划线条来提示当前页面的Tab是哪个。
2. PagerTabStrip的Tab是可以点击的，当用户点击某一个Tab时，当前页面就会跳转到这个页面，而PagerTitleStrip则没这个功能。
```

官方：PagerTabStrip是ViewPager的一个关于当前页面、上一个页面和下一个页面的一个可交互的指示器。它经常作为ViewPager控件的一个子控件被被添加在XML布局文件中。在你的布局文件中，将它作为子控件添加在ViewPager中。而且要将它的 android:layout_gravity 属性设置为TOP或BOTTOM来将它显示在ViewPager的顶部或底部。每个页面的标题是通过适配器的getPageTitle(int)函数提供给ViewPager的。

```
可以看到，除了第一句以外的其它句与PagerTitleStrip的解释完全相同。即用法也是相同的。只是PagerTabStrip是可交互的，而PagerTitleStrip是不可交互的区别。对于区别在哪些位置，即是上面的两点（是否可点击与下划线指示条）。
```

用法与PagerTitleStrip完全相同，即：

1、首先，文中提到：在你的布局文件中，将它作为子控件添加在ViewPager中。

2、第二，标题的获取，是重写适配器的getPageTitle(int)函数来获取的。

扩展：PagerTabStrip属性更改

1.更改下划线颜色：

```
pagerTabStrip = (PagerTabStrip) findViewById(R.id.pagertab);
pagerTabStrip.setTabIndicatorColorResource(R.color.green);
```

2.添加标题——重写适配器CharSequence getPageTitle(int)方法

在CharSequence getPageTitle(int position)方法返回值是，我们不返回String对象，而采用SpannableStringBuilder来构造了下包含图片的扩展String对像；

SpannableStringBuilder的使用方法，具体代码如下；

```
@Override
public CharSequence getPageTitle(int position) {

    SpannableStringBuilder ssb = new SpannableStringBuilder("
"+titleList.get(position)); // space added before text
                                        // for
    Drawable myDrawable = getResources().getDrawable(
            R.drawable.ic_launcher);
    myDrawable.setBounds(0, 0, myDrawable.getIntrinsicWidth(),
            myDrawable.getIntrinsicHeight());
    ImageSpan span = new ImageSpan(myDrawable,
            ImageSpan.ALIGN_BASELINE);

    ForegroundColorSpan fcs = new ForegroundColorSpan(Color.GREEN);// 字体颜色设置
为绿色
    ssb.setSpan(span, 0, 1, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);// 设置图标
    ssb.setSpan(fcs, 1, ssb.length(),
            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);// 设置字体颜色
    ssb.setSpan(new RelativeSizeSpan(1.2f), 1, ssb.length(),
            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    return ssb;
}
```

使用Fragment实现ViewPager滑动

1. 适配器实现——FragmentPagerAdapter

```
public class FragAdapter extends FragmentPagerAdapter {

    private List<Fragment> mFragments;

    public FragAdapter(FragmentManager fm,List<Fragment> fragments) {
        super(fm);
        // TODO Auto-generated constructor stub
        mFragments=fragments;
    }

    @Override
    public Fragment getItem(int arg0) {
        // TODO Auto-generated method stub
        return mFragments.get(arg0);
    }

    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return mFragments.size();
    }
```

这里有三个函数，根据第一部分的官方文档，可知，对于FragmentPagerAdapter的派生类，只重写getItem(int)和getCount()就可以了。

对于构造函数，这里申请了一个Fragment的List对象，用于保存用于滑动的Fragment对象，并在创造函数中初始化：

```
public FragAdapter(FragmentManager fm,List<Fragment> fragments) {
    super(fm);
    // TODO Auto-generated constructor stub
    mFragments=fragments;
}
```

2. 构造Fragment类

3. 主Activity实现

4. 可能出现的问题

问题：在MainActivity中，当写到这句：fragments.add(new Fragment1()); 向Fragment列表中添加Fragement对象实例时，会提示“无法将Fragment1()转换为fragment”

解决办法：这是因为导入包不一致，一般的问题在于：在Fragment1中导入的是android.app.Fragment， 而在这里导入类确是：android.support.v4.app.Fragment,包不同当然无法转换，统一导入为android.support.v4.app.Fragment之后就正常了.参考文章" https://blog.csdn.net/jason0539/article/details/9712273 "

小结

此前用过viewpager，用的viewpager+fragment的这种，但是，此次的学习通过阅读官方文档对viewpager有了更深的映像，并且练习了3个viewpager的demo并上传至github上；