第1章欢迎页面

打开app首先见到的是欢迎界面，欢迎页面有以下几点好处：

1、 欢迎用户使用此应用

2、 联网获取数据、初始化数据。例如检查更新，获取数据等等

3、 当获取数据的时候不至于黑屏，给用户良好的体验。

欢迎页面实现

我们要实现的功能是，如图所示，此红色背景上的小马，渐变着旋转且从小到大的展示成如上图所示的样子。

首先我们定义一个xml布局，具体代码如下，一个红色背景的相对布局，上面有一个小马图片的Imageview，我们需要实现这个小马渐变着旋转且从小到大的动画。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/rl_root_splash"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/splash_bg_newyear" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/splash_horse_newyear" />
```

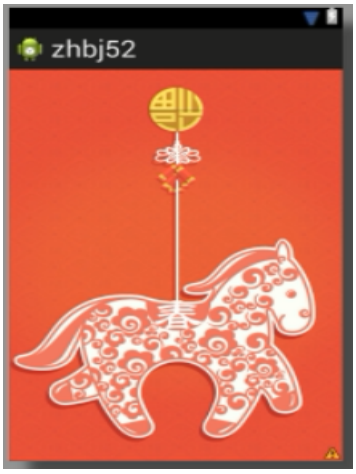
```
</RelativeLayout>
```

此动画是三个动画的合成：旋转动画、放大动画、渐变动画。代码实现入下：

```
// 动画集合
AnimationSet set = new AnimationSet(false);
// 旋转动画
RotateAnimation rotate = new RotateAnimation(0, 360,Animation.RELATIVE_TO_SELF, 0.5f,
Animation.RELATIVE_TO_SELF,0.5f);
rotate.setDuration(1000);// 动画时间
rotate.setFillAfter(true);// 保持动画状态
// 缩放动画
ScaleAnimation scale = new ScaleAnimation(0, 1, 0, 1,Animation.RELATIVE_TO_SELF, 0.5f,
Animation.RELATIVE_TO_SELF,0.5f);
scale.setDuration(1000);// 动画时间
scale.setFillAfter(true);// 保持动画状态
// 渐变动画
AlphaAnimation alpha = new AlphaAnimation(0, 1);
alpha.setDuration(2000);// 动画时间
alpha.setFillAfter(true);// 保持动画状态
set.addAnimation(rotate);
set.addAnimation(scale);
set.addAnimation(alpha);
```

全屏设置

设置完成布局之后，会出现如下图所示：



在此我们需要将闪屏界面去掉标题栏，设置成全屏模式。
目的：将欢迎界面设置成全屏模式。在清单文件中，该activity标签中增加如下属性

```
    @android:style/Theme.NoTitleBar.Fullscreen
```

引导页面

引导页面是为了告诉用户此APP有哪些好玩的或者比较好的功能，引导用户快速上手。比较扯的APP会在其中嵌入广告。一般引导页面只会在应用第一次启动的时候显示一次。

如下图所示效果图：

分析效果图如何做

如上图所示，我们需要在一个相对布局上，加上一个viewpager，一个button，一个线性布局，作用分别为：

1□viewpager用来滑动引导页。

2□Button用来放在最后一页，开始体验，打开app。

3□一个线性布局，是用来里面放入几个点，用来表明当前是哪个页面。

4□为开始体验button设置一个selector，目的是使按下按钮和没有按下时候有一个不一样的效果。

具体实现步骤

1□判断是否是第一次进入，如果是第一次进入应用，则进入引导页面
2□编写此界面的xml布局文件
3□给滑屏的viewpager设置adapter
4□为页面下方添加小圆点
5□处理实现时候具体问题

如何判断是否是第一次进入应用

思路：可以在SharedPreference中保存一个boolean变量，此变量控制是否显示引导页面。

- 步骤1 : 可以写一个缓存类，CacheUthils

```
public class CacheUtils {
	public static final String PREF_NAME = "config";
	/**
	 * 设置缓存 key 是url, value是json
	 */
	public static void setCache(String key, String value,
Context ctx) {
		SharedPreferences sp =
ctx.getSharedPreferences(PREF_NAME,
				Context.MODE_PRIVATE);
		sp.edit().putString(key, value).commit();
		//可以将缓存放在文件中, 文件名就是Md5(url), 文件内容是json
	}
	/**
	 * 获取缓存 key 是url
	 */
	public static String getCache(String key, Context ctx) {
		SharedPreferences sp =
ctx.getSharedPreferences(PREF_NAME,
				Context.MODE_PRIVATE);
		return sp.getString(key, null);
	}
}
```

- 步骤2: 动画

```
// 设置动画监听
		set.setAnimationListener(new AnimationListener() {

			// 动画执行结束
			@Override
			public void onAnimationEnd(Animation animation)
{
```

```
                        boolean userGuide =
PrefUtils.getBoolean(this, "is_user_guide_showed",false);
                        if (!userGuide) {
                              // 跳转到新手引导页
                              startActivity(new
Intent(SplashActivity.this, GuideActivity.class));
                        } else {
                              startActivity(new
Intent(SplashActivity.this, MainActivity.class));
                        }
                        finish();
                  }
                  @Override
                  public void onAnimationStart(Animation
animation) {
                  }
                  @Override
                  public void onAnimationRepeat(Animation
animation) {
                  }
            });
```

引导界面的实现

布局实现：我们这里用相对布局，布局代码如下：

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <android.support.v4.view.ViewPager
        android:id="@+id/vp_guide"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="60dp"
        android:background="@drawable/btn_guide_selector"
        android:padding="5dp"
        android:text="开始体验"
        android:visibility="invisible"
```

```
        android:textColor="@drawable/btn_guide_text_selector" />
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp" >
        <LinearLayout
            android:id="@+id/ll_point_group"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >
        </LinearLayout>
        <View
            android:id="@+id/view_red_point"
            android:layout_width="10dp"
            android:layout_height="10dp"
            android:background="@drawable/shape_point_red" />
    </RelativeLayout>
</RelativeLayout>
```

可以看出，在此布局中有一个viewpager，可以左右滑动展示。底部有个进入应用的按钮。在按钮下方有几个小点，起到索引作用。Viewpager的适配代码如下：

```
    class GuideAdapter extends PagerAdapter {
            @Override
            public int getCount() {
                  return mImageIds.length;
            }
            @Override
            public boolean isViewFromObject(View arg0, Object
arg1) {
                  return arg0 == arg1;
            }
            @Override
            public Object instantiateItem(ViewGroup container, int
position) {
                  container.addView(mImageViewList.get(position));
                  return mImageViewList.get(position);
            }
            @Override
            public void destroyItem(ViewGroup container, int
position, Object object) {
                  container.removeView((View) object);
            }
```

```
    }
```

注意：agerAdapter的destoryItem方法的super需要去掉，因为父类的代码中会抛出异常：

```
   public void destroyItem(View container, int position, Object
object) {
      throw new UnsupportedOperationException("Required method
destroyItem was not overridden");
   }
```

设置开始体验Button的状态选择器：

如命名为：btn_guide_selector，代码如下：

```
<selector
      xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:drawable="@drawable/button_red_pressed"
      android:state_pressed="true"/>
   <item android:drawable="@drawable/button_red_normal"/>
</selector>
```

初始化数据并添加底部索引小圆点：

```
   private void initViews() {
            mImageViewList = new ArrayList<ImageView>();
            // 初始化引导页的3个页面
            for (int i = 0; i < mImageIds.length; i++) {
                  ImageView image = new ImageView(this);
                  image.setBackgroundResource(mImageIds[i]);//
设置引导页背景
                  mImageViewList.add(image);
            }
            // 初始化引导页的小圆点
            for (int i = 0; i < mImageIds.length; i++) {
                  View point = new View(this);

      point.setBackgroundResource(R.drawable.shape_point_gray);//
设置引导页默认圆点
                  LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(
                              DensityUtils.dp2px(this, 10),
DensityUtils.dp2px(this, 10));
                  if (i > 0) {
                        params.leftMargin =
DensityUtils.dp2px(this, 10);// 设置圆点间隔
                  }
                  point.setLayoutParams(params);// 设置圆点的大小
                  llPointGroup.addView(point);// 将圆点添加给线性布局
            }
```

```
            // 获取视图树, 对layout结束事件进行监听

      llPointGroup.getViewTreeObserver().addOnGlobalLayoutListener
(
                        new OnGlobalLayoutListener() {
                              // 当layout执行结束后回调此方法
                              @Override
                              public void onGlobalLayout() {

      llPointGroup.getViewTreeObserver()

      .removeGlobalOnLayoutListener(this);
                                    mPointWidth =
llPointGroup.getChildAt(1).getLeft()
                                                -
llPointGroup.getChildAt(0).getLeft();
                              }
                        });
      }
```

小圆点的滑动处理

思路：除了原本的3个灰点，需要另外添加一个红色小点，默认覆盖第一个灰点，然后监听手势滑动，计算偏移量，滚动红色小点。布局，在原本的线形布局外层添加一个相对布局，然后在相对布局里添加一个红色小点。

如何计算偏移量：计算出滚动时滚动距离占屏幕宽的的百分比，乘以两点之间的距离，就可以得出小红点需要滚动的距离。

动态计算两点间的距离：


```
// 获取视图树, 对layout结束事件进行监听

      llPointGroup.getViewTreeObserver().addOnGlobalLayoutListener
(
                        new OnGlobalLayoutListener() {
                              // 当layout执行结束后回调此方法
```

```
                                @Override
                                public void onGlobalLayout() {
                                      System.out.println("layout
结束");

      llPointGroup.getViewTreeObserver()

      .removeGlobalOnLayoutListener(this);
                                      mPointWidth =
llPointGroup.getChildAt(1).getLeft()
                                                        -
llPointGroup.getChildAt(0).getLeft();
                                }
                           });
```

说明：addOnGlobalLayoutListener添加一个全局监听：说白了就是当loading这个view加载完成之后，测量完成之后，触犯这个监听器，做一些操作。往往用来获取控件宽高，后一句是移除该监听。因为知道已经测量完，监听没用了。

如何滚动呢？可以咯viewpager添加pagechangeListener监听，在此监听的onPageScrolled方法中可以实时的去获取滑动的距离。

当滑动到viewpager最后一个页面的时候，开始体验 的按钮这时候设置属性

```
btnStart.setVisibility(View.VISIABLE);
```

View的简单绘制流程

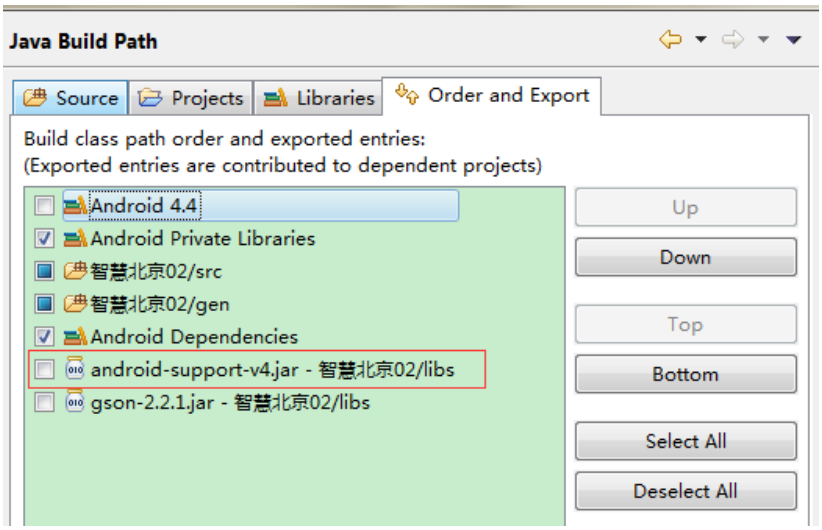
measure -> layout -> draw 先测量-再布局-再画。所以在获取小红点的左边距，需要在界面测量完之后才能获取到具体数值.

可能会出现的问题

```
 java.lang.NoClassDefFoundError: com.itheima41.zhbj.GuideUI
at
com.itheima41.zhbj.WelcomeUI$MyAnimationListener.onAnimationEnd
```

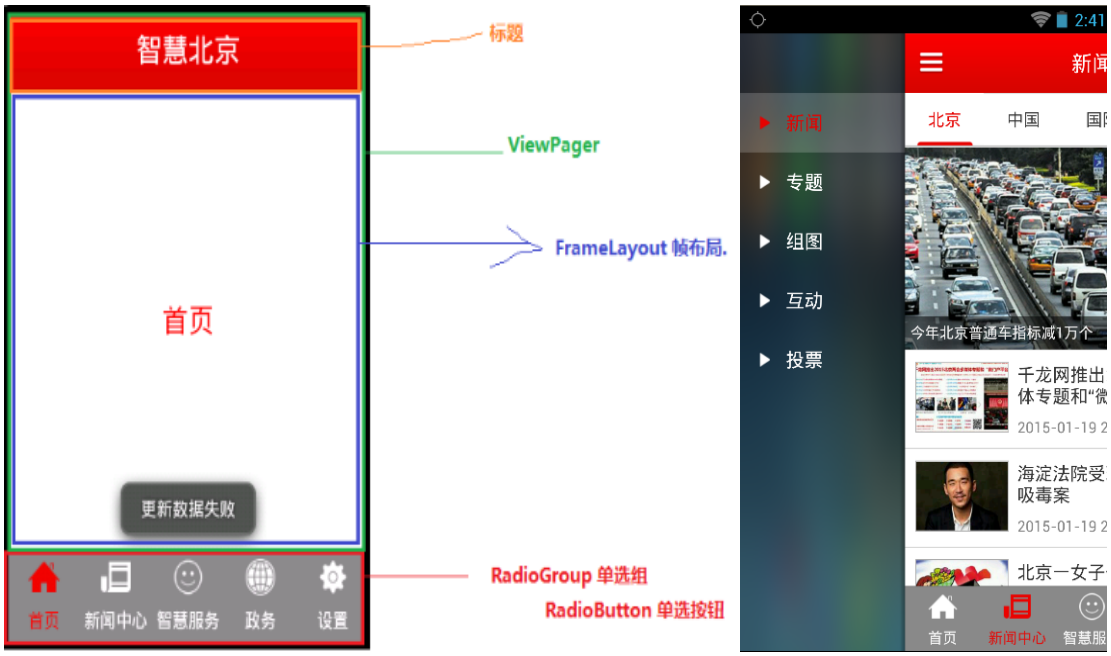
解决方法:

勾选红色方框内的jar包。如果勾选的话，打包的时候会把此jar打包到APK中。



主界面

如下图所示。



从上图可以看看出，智慧北京采用了侧滑的操作方式。在我们的项目中用到了一个开源的项目 SlidingMeun实现了侧滑。

在android的v4包中提供了一个新的DrawLayout控件，此控件也可以实现类似的功能。例如现在网易新闻的侧滑。

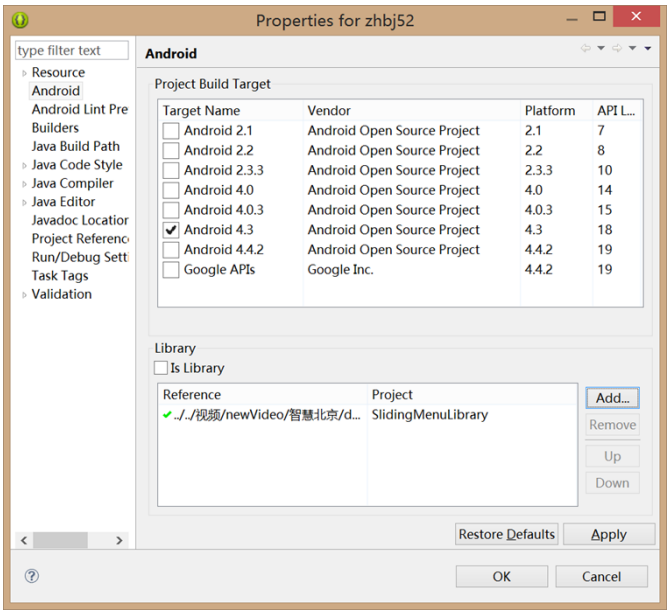
主机面的开发流程步骤

1. 找到开源项目SlidingMenu，实现侧滑栏结构；
2. 将左侧的侧滑栏Fragment实现；

3. 将底部导航栏用Radiogroup中放置RadioButton实现；
4. 顶部导航栏左侧按钮的实现。

开源项目SlidingMenu

我们把slidingmenu以一个工程的形式引入到我们的工程中。步骤如下：
1、 把函数库引入到自己的工程中. 引入方法如下图所示
右键工程---- -> properties------>左侧android选项，然后点击右下角add按钮，添加工程



2、把Activity的继承关系改为: 继承SlidingFragmentActivity, 并且需要把onCreate的修饰词修改为public.
3、设置主界面布局, 左侧菜单布局, 右侧菜单布局.
4、设置菜单相关参数: 滑动模式, 滑动范围, 主界面留在屏幕上的边界

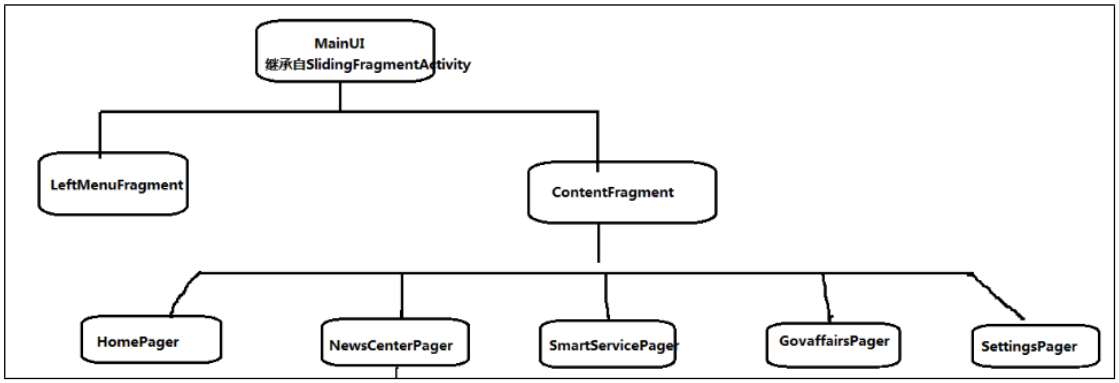
具体代码如下：

```
    public class MainActivity extends SlidingFragmentActivity {
      private static final String FRAGMENT_LEFT_MENU =
"fragment_left_menu";
      private static final String FRAGMENT_CONTENT =
"fragment_content";
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(R.layout.activity_main);
            setBehindContentView(R.layout.left_menu);// 设置侧边栏
            SlidingMenu slidingMenu = getSlidingMenu();//
获取侧边栏对象
```

```
    // 设置全屏触摸
      slidingMenu.setTouchModeAbove(SlidingMenu.TOUCHMODE_FULLSCRE
EN);
    // 获取屏幕宽度
      int width =
getWindowManager().getDefaultDisplay().getWidth();
      slidingMenu.setBehindOffset(width * 200 / 320);//
设置预留屏幕的宽度
      }
}
```

将SlidingMenu中的内容布局和左侧菜单布局用fragment代替

思路：编写一个基类BaseFragment继承fragment，然后再写两个fragment继承BaseFragment，分别为ContentFragment显示内容，LeftMenuFragment左侧菜单。



BaseFragment代码如下：

```
    public abstract class BaseFragment extends Fragment {
      public Activity mActivity;
      // fragment创建
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            mActivity = getActivity();
      }
      // 处理fragment的布局
      @Override
      public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                  Bundle savedInstanceState) {
            return initViews();
      }
      // 依附的activity创建完成
```

```
	@Override
	public void onActivityCreated(Bundle savedInstanceState) {
		super.onActivityCreated(savedInstanceState);
		initData();
	}
	// 子类必须实现初始化布局的方法
	public abstract View initViews();
	// 初始化数据，可以不实现
	public void initData() {
	}
}
```

Fragment替换代码

```
FragmentManager fm = getSupportFragmentManager();
// 开启事务
FragmentTransaction transaction = fm.beginTransaction();
// 用fragment替换framelayout
transaction.replace(R.id.fl_left_menu,
LeftMenuFragment(),FRAGMENT_LEFT_MENU);
transaction.replace(R.id.fl_content, new
ContentFragment(),FRAGMENT_CONTENT);
// 提交事务
transaction.commit();
```

底部导航为RadioGroup

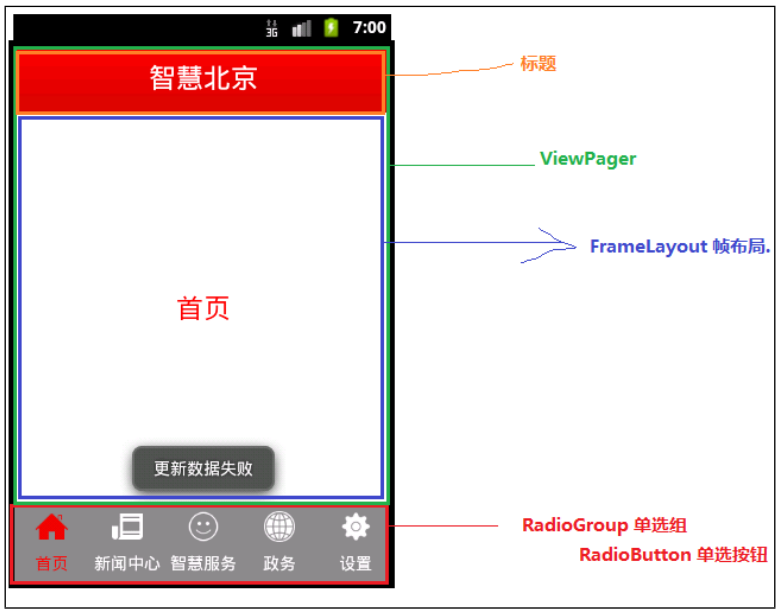
底部导航按钮的选择器：

```
<selector
  xmlns:android="http://schemas.android.com/apk/res/android">
  <itemandroid:drawable="@drawable/home_press"android:state_checke
  d="true"/>
   <item android:drawable="@drawable/home"/>
</selector>
```

RadioButton如果不添加id，RadioGroup是可以选中多个的，并且RadioButton是无法取消选择的。

内容显示区域ContentFragment

内容显示区域如下图所示：



顶部标题栏+中间显示内容区域viewpager，一共5个pager，分别对应着首页、新闻中心、智慧服务、政务及设置。

在显示内容区域的时候，我们放弃了使用setContentView()方法。(其实底层还是调用了此方法)这个地方用到了xUtil第三方工具。

```
@ViewInject(R.id.vp_content)
private ViewPager mViewPager;
@Override
public View initViews() {
      View view = View.inflate(mActivity,
R.layout.fragment_content, null);
      // 注入view和事件，替代了SetContentView
      ViewUtils.inject(this, view);
 return view;
}
```

上面标题栏用相对布局实现，中间内容区域用帧布局。因为每个页面都是用这种方式实现的，所以我们写一个基类TabBasePager以达到复用的效果。代码如下：

```
public class TabBasePager {
  public Activity mActivity;
  public View mRootView;// 布局对象
  public TextView tvTitle;// 标题对象
  public FrameLayout flContent;// 内容
  public ImageButton btnMenu;// 菜单按钮
  public ImageButton btnPhoto;// 组图切换按钮
  public BasePager(Activity activity) {
      mActivity = activity;
      initViews();
  }
```

```
	/**
	 * 初始化布局
	 */
	public void initViews() {
		mRootView = View.inflate(mActivity,
R.layout.base_pager, null);
		tvTitle = (TextView)
mRootView.findViewById(R.id.tv_title);
		flContent = (FrameLayout)
mRootView.findViewById(R.id.fl_content);
		btnMenu = (ImageButton)
mRootView.findViewById(R.id.btn_menu);
		btnPhoto = (ImageButton)
mRootView.findViewById(R.id.btn_photo);
		btnMenu.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				toggleSlidingMenu();
			}
		});
	}
	/**
	 * 切换SlidingMenu的状态
	 *
	 * @param b
	 */
	protected void toggleSlidingMenu() {
		MainActivity mainUi = (MainActivity) mActivity;
		SlidingMenu slidingMenu = mainUi.getSlidingMenu();
		slidingMenu.toggle();// 切换状态, 显示时隐藏, 隐藏时显示
	}
	/**
	 * 初始化数据
	 */
	public void initData() {
	}
	/**
	 * 设置侧边栏开启或关闭
	 *
	 * @param enable
	 */
	public void setSlidingMenuEnable(boolean enable) {
		MainActivity mainUi = (MainActivity) mActivity;
		SlidingMenu slidingMenu = mainUi.getSlidingMenu();
```

```
            if (enable) {

        slidingMenu.setTouchModeAbove(SlidingMenu.TOUCHMODE_FULLSCRE
EN);
            } else {

        slidingMenu.setTouchModeAbove(SlidingMenu.TOUCHMODE_NONE);
            }
        }
}
```

中间内容区域的实现：在中间显示区域的viewpager添加一个适配器。View page的每一个页面都是继承了BasePager。

当我们点击下面的RadioButton的时候，我们为了不让内容区域有viewpager的滑动效果及预加载效果，我们这里自定义一个Viewpage类，叫做NoScrollViewPager。此类继承Viewpager，

禁止滑动：重写onInterceptTouchEvent和onTouchEvent，都返回false。底部导航切换：mViewPager.setCurrentItem(0)调用此方法实现切换页面

顶部左侧导航菜单:

在顶部栏左侧添加按钮，点击按钮打开左侧菜单，代码如下：

```
@Override
    public void onClick(View v) {
        MainActivity mainUi = (MainActivity) mActivity;
        SlidingMenu slidingMenu = mainUi.getSlidingMenu();
    // 切换状态, 显示时隐藏, 隐藏时显示
        slidingMenu.toggle();
}
```