第2章 服务器的搭建

搭建步骤

把zhbj的文件夹(服务端工程)扔进apache-tomcat-6.0.36\webapps\ROOT目录下, 在回到bin目录startup.bat, 开启服务.

检测服务器是否可以正常访问

在浏览器中输入: http://localhost:8080/zhbj/categories.json。针对于我们模拟器的话，localhost是访问不到的；

需要我们将url地址设置为：http://10.0.2.2:8080/zhbj/categories.json如果能够正常访问，并获取到数据，说明服务器没有问题。数据是一个json字符串，格式化之后如下图：

左侧菜单的实现

如下图所示，既左侧菜单

左侧菜单的数据是来自网络的，要想获取左侧菜单，我们需要联网获取菜单数据。

网络访问

在网络访问的时候，有好多种方式，可以用apache组织提供的httpClient对象，也可以用谷歌提供的HttpUrlConneet对象。当然也可以使用第三方封装好的请求对象。在此我们使用了xUtils中的HttpUtils类来访问网络。

在NewsCenterPager中添加如下代码：

```
/**
	 * 从服务器获取数据
	 */
	private void getDataFromServer() {
		HttpUtils utils = new HttpUtils();
		// 使用xutils发送请求
		utils.send(HttpMethod.GET,
GlobalContants.CATEGORIES_URL,
				new RequestCallBack<String>() {
					// 访问成功，在主线程运行
					@Override
					public void onSuccess(ResponseInfo
responseInfo) {
						String result = (String)
responseInfo.result;
						System.out.println("返回结果:" +
result);
						parseData(result);
						// 设置缓存
```

```
            CacheUtils.setCache(GlobalContants.CATEGORIES_URL,
                                                    result, mActivity);
                                }
                                // 访问失败, 在主线程运行
                                @Override
                                public void onFailure(HttpException
error, String msg) {
                                        Toast.makeText(mActivity, msg,
Toast.LENGTH_SHORT)
                                                    .show();
                                        error.printStackTrace();
                                }
                        });
            }
```

数据解析

现在数据传递的方式有XML和json两种形式，一般json用的比较多。智慧北京用的是json数据格式。

解析json数据有好多种方式。可以使用android自带的JsonObject对象进行解析。也可以用第三方jar进行解析。在此项目中我们使用到了谷歌提供的GSON开源项目进行解析。

要想使用Gson解析Json，首先需要先建立与Json节点相对于的JavaBean。这个JavaBean中所有字段都需要公有化public，因为Gson是通过反射来做的。

如果json中包含数组[]，那么javabean中就用集合List，list存储的对象可以在当前javabean中定义内部类。使用方法如下：

```
    protected void parseData(String result) {
            Gson gson = new Gson();
        // NewsDate为我们的javaBean,其中的字段与JSON字段对应
            mNewsData = gson.fromJson(result, NewsData.class);
        }
```

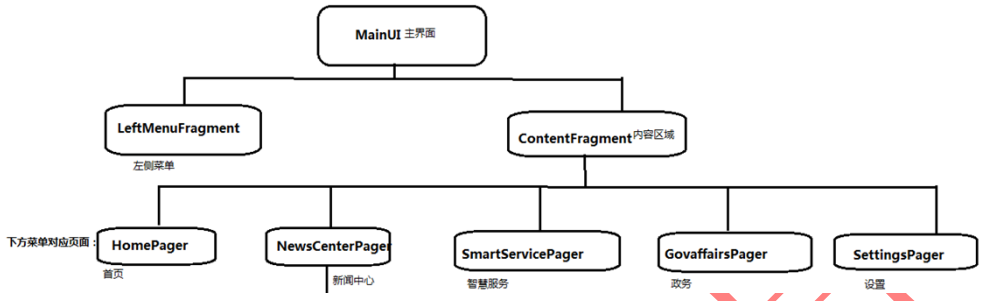
此处NewsData即为我们需要把json解析成的对象，这个是需要我们提前定义好的，根据json中的属性，来定义出一个对象Bean，定义出的Bean如下：

```
/**
 * 网络分类信息的封装
 *
 * 字段名字必须和服务器返回的字段名一致, 方便gson解析
 *
 * @author Kevin
```

```
 *
 */
public class NewsData {

	public int retcode;
	public ArrayList<NewsMenuData> data;

	// 侧边栏数据对象
	public class NewsMenuData {
		public String id;
		public String title;
		public int type;
		public String url;

		public ArrayList<NewsTabData> children;

		@Override
		public String toString() {
			return "NewsMenuData [title=" + title + ",
children=" + children
					+ "]";
		}
	}

	// 新闻页面下11个子页签的数据对象
	public class NewsTabData {
		public String id;
		public String title;
		public int type;
		public String url;

		@Override
		public String toString() {
			return "NewsTabData [title=" + title + "]";
		}
	}

	@Override
	public String toString() {
		return "NewsData [data=" + data + "]";
	}

}
```

左侧菜单的实现

获取到数据后，我们需要把数据显示到左侧菜单中，那么如何设置呢？先看下图



我们想在NewsCenterPager中获取到左侧菜单，可以通过MainUI来获取，因为leftMenuFragment和NewsCenterPager的父类都是MainUI。在MainUi中定义如下方法。

```
// 获取侧边栏fragment
      public LeftMenuFragment getLeftMenuFragment() {
            FragmentManager fm = getSupportFragmentManager();
            //这里通过findFragmentByTag方法来找到左侧菜单
            LeftMenuFragment fragment = (LeftMenuFragment) fm
                        .findFragmentByTag(FRAGMENT_LEFT_MENU);
            return fragment;
      }
```

获取到左侧菜单后，将网络访问的数据传递给菜单，在parseData()方法中添加如下代码：

```
    protected void parseData(String result) {
            // 刷新测边栏的数据
            MainActivity mainUi = (MainActivity) mActivity;
            LeftMenuFragment leftMenuFragment =
mainUi.getLeftMenuFragment();
            //setMenuData方法需要在左侧菜单实现
            leftMenuFragment.setMenuData(mNewsData);
      }
```

左侧菜单收到数据后将内容展示。其实左侧菜单就是一个listview，具体代码如下，分别为左侧菜单定义以及adapter中每一个item的xml定义

```
<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"
    android:background="#000" >

    <ListView
        android:id="@+id/lv_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:listSelector="@android:color/transparent"
        android:divider="@android:color/transparent"
        android:layout_marginTop="40dp" />

</RelativeLayout>
```

每一个item的xml定义如下：

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="10dp" >

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:drawableLeft="@drawable/btn_menu_selector"
        android:drawablePadding="5dp"
        android:enabled="false"
        android:text="新闻"
        android:textColor="@drawable/text_menu_selector"
        android:textSize="25sp" />

</LinearLayout>
```

但是有几个需要注意的地方：

因为左侧菜单是可以点击的，当点击每个item的时候，它的状态是需要改变的，变为选中的颜色，其他的item变为未选中的状态。实现如下：

```
@Override
```

```
                public void onItemClick(AdapterView<?> parent,
View view,
                        int position, long id) {
                    mCurrentPos = position;
                //刷新listview的时候，需要getView方法的配合
                    mAdapter.notifyDataSetChanged();
                //当点击左侧菜单的时候，需要切换内容区域
                    setCurrentMenuDetailPager(position);
                    toggleSlidingMenu();// 隐藏左侧菜单
                }
```

通过getView()方法设置item的颜色

```
@Override
            public View getView(int position, View convertView,
ViewGroup parent) {
                View view = View.inflate(mActivity,
R.layout.list_menu_item, null);
                TextView tvTitle = (TextView)
view.findViewById(R.id.tv_title);
                NewsMenuData newsMenuData = getItem(position);
                tvTitle.setText(newsMenuData.title);
                if (mCurrentPos == position) {//
判断当前绘制的view是否被选中
                    // 显示红色
                    tvTitle.setEnabled(true);
                } else {
                    // 显示白色
                    tvTitle.setEnabled(false);
                }
                return view;
            }
```

点击左侧菜单切换内容区域

目的：点击左侧菜单，将内容区域切换到相应的界面。在切换主界面之前，需要先创建一个基类，因为不同菜单切换的界面都类似。基类代码如下所示：

```
   public abstract class BaseMenuDetailPager {
     public Activity mActivity;
     public View mRootView;// 根布局对象
     public BaseMenuDetailPager(Activity activity) {
         mActivity = activity;
```

```
            mRootView = initViews();
        }
        /**
         * 初始化界面
         */
        public abstract View initViews();
        /**
         * 初始化数据
         */
        public void initData() {
        }
}
```

当点击左侧菜单的时候，内容显示区域切换不同的页面。在此我们创建四个不同的页面来进行点击时候切换显示的页面。

- 新闻菜单：NewsMenuPager
- 专题：TopicMenuPager
- 组图：PhotosMenuPager
- 互动：InteractMenuPager

将四个菜单添加到NewsCenterPager中。在NewsCenterPager中实例化一个NewsCenterMenuBasePager的集合，将4个界面添加进去。然后编写点击菜单切换主界面的方法switchPager，这里的主界面是指的谁呢？TabBasePager中的FrameLayout。添加代码如下：

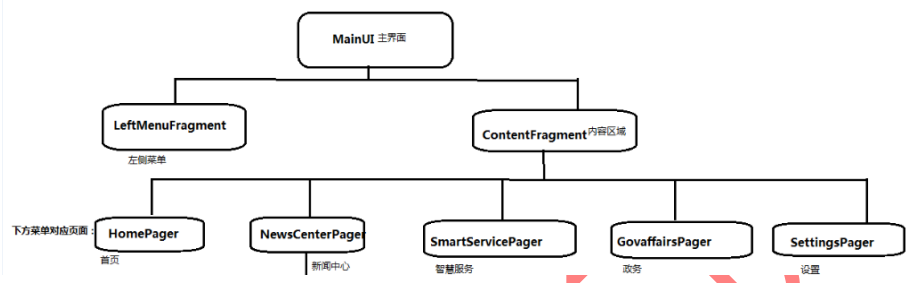
```
// 准备4个菜单详情页
            mPagers = new ArrayList<BaseMenuDetailPager>();
            mPagers.add(new NewsMenuDetailPager(mActivity,
                        mNewsData.data.get(0).children));
            mPagers.add(new TopicMenuDetailPager(mActivity));
            mPagers.add(new PhotoMenuDetailPager(mActivity,
btnPhoto));
            mPagers.add(new InteractMenuDetailPager(mActivity));
```

切换不同的页面的方法：

```
    public void switchPager(int position) {
            NewsCenterMenuBasePager pager =
pagerList.get(position);
            View view = pager.getRootView();
            //flContent就是TabBasePager中的FrameLayout
            flContent.removeAllViews();//添加之前先清空，防止界面重叠
            flContent.addView(view);
```

```
            tvTitle.setText(mLeftMenuList.get(position).title);
            pager.initData(); // 当前需要显示界面, 需要把数据先初始化了.
        }
```

如何调用switchPager切换页面：默认值和菜单listview是在LeftMenuFragment中设置的，那么LeftMenuFragment中如何调用NewsCenterPager中的方法？请看图



那么Main中需要拿到ContentFragment？

```
        /**
         * alt + shift + J 加注释
         * 获取主界面的正文fragment
         * @return
         */
        public ContentFragment getContentFragment() {
            FragmentManager fm = getSupportFragmentManager();
            ContentFragment contentFragment = (ContentFragment)
fm.findFragmentByTag(CONTENT_FRAGMENT_TAG);
            return contentFragment;
        }
```

ContentFragment如何拿到NewsCenterPager？

```
    public NewsCenterPager getNewsCenterPager() {
        return  (NewsCenterPager) pagerList.get(1);
}
```

在LeftMenuFragment中定义：

```
    private void switchNewsCenterPager() {
            MainUI mainUI = ((MainUI) mActivity);
            ContentFragment contentFragment =
mainUI.getContentFragment();
        NewsCenterPager newsCenterPager =
  contentFragment.getNewsCenterPager();
            newsCenterPager.switchPager(currentEnalbledPosition);
}
```

在初始化时调用：LeftMenuFragment.setMenuDataList(List<NewsCenterMenu>)

```
        // 设置主界面默认显示的布局
switchNewsCenterPager();
```

菜单listview item 点击监听时调用:
LeftMenuFragment.onItemClick(AdapterView<?>, View, int, long)

```
// 把菜单关闭, 显示主界面
            ((MainUI) mActivity).getSlidingMenu().toggle();
            // 把主界面切换成对应菜单的页面
            switchNewsCenterPager();
```

等到了这里，我们就完成了通过侧边栏切换主内容界面的切换，依次方法是，通过侧边栏的item点击，获取到mainactivity，通过mainactivity获取到ContentFragment，通过ContentFragment进入NewsCenterPager中，然后在NewsCenterPager中，调用switchPager方法，进行切换界面。

部分问题解决以及优化

截至目前，我们已经可以通过左侧菜单栏进行切换主界面内容了，但是存在一些小问题，或者小优化；

1□切换SlidingMenu的状态之后，需要隐藏侧边栏

i. 代码如下：

```
/**
         * 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
       }
```

2□获取完数据之后，展示数据，左侧栏需要默认选择第一条

```
setCurrentMenuDetailPager(0);// 设置菜单详情页-新闻为默认当前页
```

3□选中哪个标题，需要主界面的标题也跟着变化

a) 在basepager中获取到标题的View（tvTitle），然后在每个Pager中具体实现

```
tvTitle.setText(menuData.title);
```

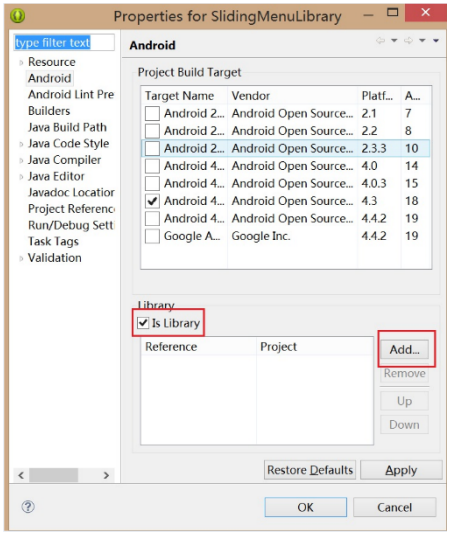
新闻中心页面的实现

点击底部导航新闻中，然后点击左侧菜单的“新闻”菜单后，展示界面如下:

分析此界面得知：此界面可分为两部分，一层为viewpagerindicator，还有一层就是绑定的viewpager，如下图所示：

顶部为ViewPagerIndicator，指示器，用来展示新闻分类。底部是ViewPager，用来展示新闻列表。ViewPagerIndicator为一个开源项目，可以在github上下载，下载地址：https://github.com/JakeWharton/ViewPagerIndicator下载完成后将项目导入开发工具，然后我们的项目引用此项目，然后就可以使用了。

一个项目如何引用另外一个项目:右键要导入的工程----->properties------>Android出现如下图所示：



在Is Library前面的复选框搭上对勾，打上对勾的意思是此工程作为一个库类，可以让其他工程引用，并调用里面的方法。那别的工程怎么引用呢？
打开上图窗口，步骤和上面一样，然后点击右下角的Add按钮，添加我们需要的工程即可。

- 集成ViewPagerIndicator

顶部指示器布局，横向的线形布局，然后TabPageIndicator的宽的权重是1，箭头的宽度是warp_content.这样的话，就是除了箭头占的空间，剩下的都给TabPageIndicator了。布局如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <com.viewpagerindicator.TabPageIndicator
            android:id="@+id/tpi_news_menu"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
        <ImageButton
            android:id="@+id/ib_news_menu_next_tab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
```

```
            android:layout_marginLeft="10dip"
            android:layout_marginRight="10dip"
            android:background="@android:color/transparent"
            android:src="@drawable/news_cate_arr" />
    </LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vp_news_menu"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

Java代码中实例化我们使用了xUtils的方法：

```
@Override
      public View initViews() {
            View view = View.inflate(mActivity,
R.layout.news_menu_pager, null);
            ViewUtils.inject(this, view); // 注入view和事件
            return view;
      }
```

- **符合需求的viewpagerindicator**
  - 通过看源码我们发现，指示器中有个自带的style，此style代码如下，关于背景，有个默认的drawable，是一个状态选择器

```
<style name="Widget.TabPageIndicator" parent="Widget">
    <item name="android:gravity">center</item>
    <item name="android:background">@drawable/vpi__tab_indicator</item>
    <item name="android:paddingLeft">22dip</item>
    <item name="android:paddingRight">22dip</item>
    <item name="android:paddingTop">12dp</item>
    <item name="android:paddingBottom">12dp</item>
    <item name="android:textAppearance">@style/TextAppearance.TabPageIndicator</item>
    <item name="android:textSize">16sp</item>
    <item name="android:textColor">#000</item>
    <item name="android:maxLines">1</item>
</style>
```

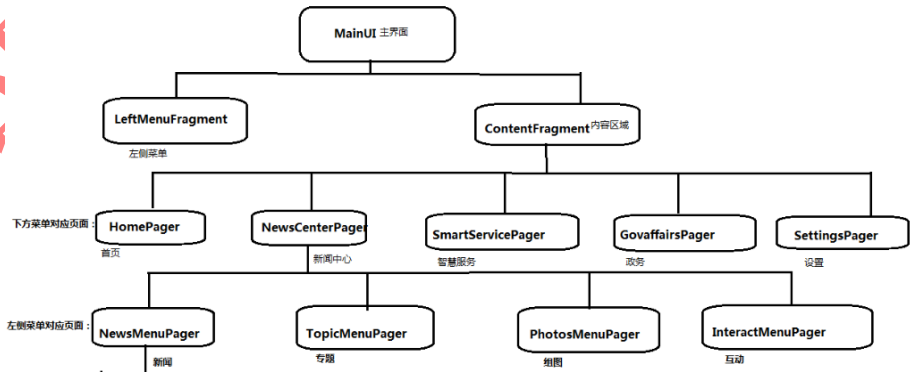
  - 状态选择器代码如下，我们需要更改此状态选择器中的背景图片，来满足我们的需求：

```
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Non focused states -->
    <item android:state_focused="false"
android:state_selected="false" android:state_pressed="false"
android:drawable="@android:color/transparent" />
    <item android:state_focused="false"
android:state_selected="true"  android:state_pressed="false"
android:drawable="@drawable/news_tab_item_bg_select" />

    <!-- Focused states -->
```

```
    <item android:state_focused="true"
android:state_selected="false" android:state_pressed="false"
android:drawable="@android:color/transparent" />
    <item android:state_focused="true"
android:state_selected="true"  android:state_pressed="false"
android:drawable="@drawable/news_tab_item_bg_select" />

    <!-- Pressed -->
    <!--    Non focused states -->
    <item android:state_focused="false"
android:state_selected="false" android:state_pressed="true"
android:drawable="@android:color/transparent" />
    <item android:state_focused="false"
android:state_selected="true"  android:state_pressed="true"
android:drawable="@drawable/news_tab_item_bg_select" />

    <!--    Focused states -->
    <item android:state_focused="true"
android:state_selected="false" android:state_pressed="true"
android:drawable="@android:color/transparent" />
    <item android:state_focused="true"
android:state_selected="true"  android:state_pressed="true"
android:drawable="@drawable/news_tab_item_bg_select" />
</selector>
```

- **指示器的数据**

关于指示器代码应该写在哪里，我们再来看一下结构图：



我们发现在NewsMenuPager类中添加指示器的代码。指示器的数据哪里来？

在NewsCenterPager中获取左侧菜单数据时，已经获取到了新闻相关的类型。所以可以在实例化NewsMenuPager时传递进来。查看传递代码processData方法 (如果当时没有传递，那么现在可以添加参数传递)。

新闻中心页面的数据实体类：NewsCenterMenuBean中的children就是指示器页签数据。构造函数代码入下：

```
    private ArrayList<NewsTabData> mNewsTabData;// 页签网络数据
    public NewsMenuPager(Context context, NewsCenterMenu
newsCenterMenu) {
        super(context);
        tabBeanList = newsCenterMenu.children;
    }
```

- viewpager适配器（适配器的其他方法可以暂时不重写）

```
    /**
         * 重写此方法,返回页面标题,用于viewpagerIndicator的页签显示
         */
        @Override
        public CharSequence getPageTitle(int position) {
            return mNewsTabData.get(position).title;
        }
```

- 给ViewPagerIndicator传递数据

就通过以下代码，传递过去viewpager以后，那么viewpager里的数据是从pageAdapger中拿的，上边用了一个新方法getPageTitle，这个方法就是指定指示器标签的内容的。

```
mIndicator.setViewPager(mViewPager);
```

顶部指示器样式：样式需要给一个Activity添加，而NewsMenuPager不是一个Activity，无法指定主题，那么就可以向上找-MainUI。所以只需要给MainUI知道主题即可。

```
<activity android:name="com.itheima41.zhbj.MainUI"
        android:theme="@style/Theme.PageIndicatorDefaults"
></activity>
```

这个样式是ViewPagerIndicator第三方库提供的样式。

- 当我们把指示器嵌套进去之后，指示器的问题就显露出来，指示器可以左右滑动，但是定义的策划菜单也可以左右滑动，这样一来，作为滑动事件就产生了冲突，也就是说，安卓系统不知道我们需要滑动的是哪个，就有问题了。
- 指示器与SlidingMenu滑动冲突

1. 问题描述

有可能出现在指示器内容区域左滑，SlidingMenu菜单滑出来了，而指示器并没有动，这是因为SlidingMenu抢走了指示器的事件

2. 分析

SlidingMenu是MainUI里的组件，指示器属于NewsMenuPager类中，他俩有父子层级关系，现在子类要阻止父类处理事件

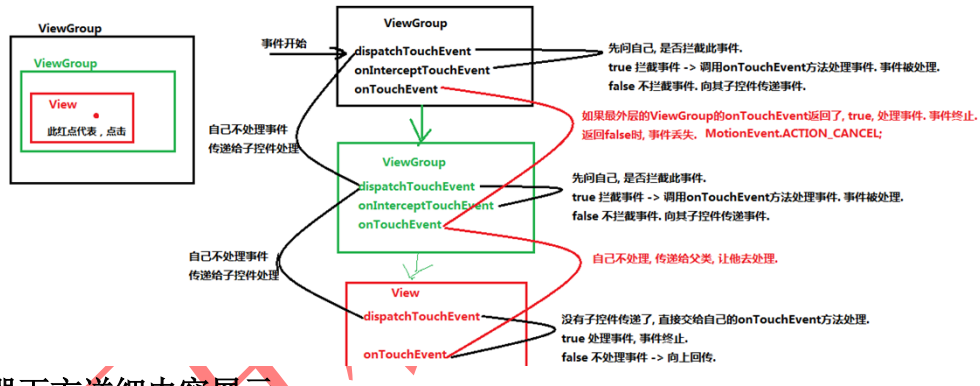
3. dispatchTouchEvent：分发事件

可以申请不让父控件拦截我的事件getParent().requestDisallowInterceptTouchEvent(true);设置不允许拦截事件，所以只需要在ViewPagerIndicator中调用此方法就行了。

4. onInterceptTouchEvent：拦截事件

返回true：拦截事件--
调用onTouchEvent方法处理事件，事件被处理，结束。返回false：不拦截事件，向其子控件传递事件。

5. onTouchEvent：处理事件
6. 事件传递机制图



➢ **指示器下方详细内容展示**

1. 详情页

有11个页签，每个界面都一样，那么我们就需要定义一个类即NewsMenuTabDetailPager。因为一些公共方法都一样，所以这个类就继承NewsCenterMenuBasePager即可。具体代码如下：

```
/**
 * @author andong
 * 新闻中心页签对应的新闻列表界面
 */
   public class NewsMenuTabDetailPager extends
NewsCenterMenuBasePager {

   public NewsMenuTabDetailPager(Context context) {
            super(context);
      }
@Override
      public View initView() {
            return null;
      }
```

```
}
```

2. 给NewsMenuPager添加页签详情页NewsMenuTabDetailPager

页签数据已经传递过来，那么我们需要用这个数据来实例化页签详情页。代码如下：

```
@Override
        public void initData() {
                // 初始化ViewPager数据.
                tabDetailPagerList = new ArrayList<NewsMenuTabDetailPager>();
                for (int i = 0; i < tabBeanList.size(); i++) {
                        tabDetailPagerList.add(new NewsMenuTabDetailPager(mContext,
tabBeanList.get(i)));//这里的第二个参数，就是页签数据NewsCenterTabBean，他有个属性是url，这个url就是请求此页签具体数据的url，在后边会用到。
                }
                NewsMenuAdapter mAdapter = new NewsMenuAdapter();
                mViewPager.setAdapter(mAdapter);
                // 把ViewPager和页签关联.
                mIndicator.setViewPager(mViewPager);
                // 监听页面的改变
(此方法可以暂时不写，为以后解决轮播图滑动与外层Viewpager滑动冲突而添加)
                mIndicator.setOnPageChangeListener(this);
        }
```

3. NewsMenuAdapter的代码如下：

```
    class NewsMenuAdapter extends PagerAdapter {
                @Override
                public int getCount() {
                        return tabDetailPagerList.size();
                }
                @Override
                public boolean isViewFromObject(View arg0, Object
arg1) {
                        return arg0 == arg1;
                }
                @Override
                public void destroyItem(ViewGroup container, int
position, Object object) {
                        container.removeView((View) object);
                }
                /**
                 * 返回的String会作为TabPageIndicator的页签数据来展示，
并且和对应position位置的页面想匹配
                 */
                @Override
                public CharSequence getPageTitle(int position) {
                        return tabBeanList.get(position).title;
```

```
            }
            @Override
            public Object instantiateItem(ViewGroup container, int
position) {
                    NewsMenuTabDetailPager pager =
tabDetailPagerList.get(position);
                    View view = pager.getRootView();
                    container.addView(view);
                    pager.initData();//调用界面初始化数据方法
                    return view;
            }
      }
```

4. 界面布局

整体布局：顶部轮播图，viewpager，下边是一个listview。轮播图布局：外层相对布局，上边是viewpager，下边是一个内层相对布局包括一个textview、一个LinearLayout来放轮播图的指示器。代码如下：tab_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
      <!-- 轮播图+右下角的指示器 -->
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="185dip" >
            <!-- 轮播图，viewpager -->
        <android.support.v4.view.ViewPager
            android:id="@+id/vp_tab_detail_topnews"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" >
        </android.support.v4.view.ViewPager>
            <!-- 指示器，相对布局 -->
        <RelativeLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:background="#33000000"
            android:paddingBottom="5dip"
            android:paddingLeft="10dip"
            android:paddingRight="10dip"
            android:paddingTop="5dip" >

            <TextView
```

```
                android:id="@+id/tv_tab_detail_description"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="默认的描述信息"
                android:textColor="#FFFFFF" />

            <LinearLayout
                android:id="@+id/ll_tab_detail_point_group"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:orientation="horizontal" >
            </LinearLayout>
        </RelativeLayout>
    </RelativeLayout>
      <!-- 新闻列表，listView -->
    <ListView
        android:id="@+id/lv_tab_detail_news"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
    </ListView>
</LinearLayout>
```

5. 代码中：NewsMenuTabDetailPager

```
@Override
      public View initView() {
            View view = View.inflate(mContext,
R.layout.tab_detail, null);
            ViewUtils.inject(this, view);
            return view;
      }
```

6. 轮播图数据填充

访问网络，获取数据。页签请求的url，已经通过构造传递过来。代码如下：

```
@Override
   public void initData() {
         String url = Constants.SERVICE_URL +
mNewsCenterTabBean.url;
         //请求网络
         getDataFromNet(url);
   }
   private void getDataFromNet(final String url) {
        //先取缓存
         String json = CacheUtils.getString(mContext, url,
null);
```

```
        if(!TextUtils.isEmpty(json)) {
              processData(json);
        }
        //访问网络
        HttpUtils utils = new HttpUtils();
        utils.send(HttpMethod.GET, url, new
RequestCallBack<String>() {
              @Override
        public void onSuccess(ResponseInfo<String>
responseInfo) {
                    System.out.println(mNewsCenterTabBean.title
+ "数据请求成功: " + responseInfo.result);
                    CacheUtils.putString(mContext, url,
responseInfo.result);//缓存
                    // 缓存完数据, 开始处理数据.
                    processData(responseInfo.result);
              }
              @Override
              public void onFailure(HttpException error,
String msg) {
                    System.out.println(mNewsCenterTabBean.title
+ "数据请求失败: " + msg);
              }
    ));
)
```

7. 请求获取到数据的处理

在请求到数据后，我们可以根据json数据的格式来写出对应bean对象。在此解析数据方面我们用到了谷哥提供的Gson包来进行解析。解析数据代码如下：

```
/**
      * 接收json数据, 进行解析, 并展示界面.
      * @param result
      */
     protected void processData(String result) {
           Gson gson = new Gson();
           TabDetailBean bean = gson.fromJson(result,
TabDetailBean.class);

           // 初始化顶部轮播新闻的数据
           topNewsList = bean.data.topnews;
           //顶部轮播图的适配器, 下边会创建此适配器
           TopNewsAdapter mAdapter = new TopNewsAdapter();
           topNewViewPager.setAdapter(mAdapter);
     }
```

8. 顶部轮播图适配器

问题：给ImageView指定src，src是会保留图片原来的比例，有可能上边或下边有空白，没有填充满控件，如何解决？可以通过设置ScaleType为Fix_xy来让图片填充控件，fix_xy会拉伸图片让其充满。

问题：图片是从网络获取的，有可能图片没拿到，所以需要给设置一个默认图片。代码如下：

```
    class TopNewsAdapter extends PagerAdapter {
            @Override
            public int getCount() {
                  return topNewsList.size();
            }
            @Override
            public boolean isViewFromObject(View arg0, Object
arg1) {
                  return arg0 == arg1;
            }
            @Override
            public void destroyItem(ViewGroup container, int
position, Object object) {
                  container.removeView((View) object);
            }
            @Override
            public Object instantiateItem(ViewGroup container, int
position) {
                  ImageView iv = new ImageView(mContext);
                  iv.setScaleType(ScaleType.FIT_XY);
                  // 设置一张默认的图片

      iv.setImageResource(R.drawable.home_scroll_default);
                  container.addView(iv); //
把ImageView添加到ViewPager中.
                  // 请求网络图片，把图片展示给ImageView
                  TopNew topNew = topNewsList.get(position);
                  bitmapUtils.display(iv, topNew.topimage);
          //这个bitmapUtils在构造中实例化
                  return iv;
            }
      }
```

9. 实例化bitmapUtils

```
    //通过BitMapUtils来设置网络图片
      bitmapUtils = new BitmapUtils(mContext);
    bitmapUtils.configDefaultBitmapConfig(Config.ARGB_4444);
```

➢ 新闻切换的ViewPager与SlidingMenu的冲突如何解决？

滑到第一个新闻页签的时候，左滑可以拖出SlidingMenu菜单。解决：在NewsM

enuPager中给Viewpager添加setOnPageChangeListener页面切换监听，在onPageSelected判断如果是第一个，那么就设置左侧菜单可以拖动，如果不是第一个，那就不可以拖动。

```
@Override
      public void onPageSelected(int position) {
            // 如果position的位置是0, 菜单置为可用.
            if(position == 0) {
                  ((MainUI)
mContext).getSlidingMenu().setTouchModeAbove(SlidingMenu.TOUCHMOD
E_FULLSCREEN);
            } else {
                  ((MainUI)
mContext).getSlidingMenu().setTouchModeAbove(SlidingMenu.TOUCHMOD
E_NONE);
            }
      }
```

viewpagerindicator的使用总结

ViewPager指针项目，在使用ViewPager的时候能够指示ViewPager所在的位置，就像Google
Play中切换的效果一样，还能使用在应用初始化的介绍页面，具体步骤如下：
1. 引入ViewPagerIndicator库
2. 编写布局文件

```
 <com.viewpagerindicator.TabPageIndicator
        android:id="@+id/indicator"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
```

3.
mIndicator.setViewPager(mViewPager);//将viewpager和mIndicator关联起来,必须在viewpager设置完adapter后才能调用

4. 重写PagerAdapter方法,返回页面标题

```
/**
             * 重写此方法,返回页面标题,用于viewpagerIndicator的页签显示
             */
            @Override
            public CharSequence getPageTitle(int position) {
                  return mNewsTabData.get(position).title;
            }
```

5. 自定义样式修改