


## 第3章顶部轮播图

### 指示器右侧箭头

如图：逻辑：点击箭头，页签切换到下一个

实现：利用xUtils给按钮添加点击事件。

```
@OnClick(R.id.ib_news_menu_next_tab)
public void nextTab(View v) {
    mViewPager.setCurrentItem(mViewPager.getCurrentItem()
+ 1);
}
```

### 轮播图ViewPager与外层新闻种类ViewPager冲突

思路：不让父元素拦截子控件的事件，需要在子控件的分发事件的方法dispatchTouchEvent中调用:getParent().requestDisallowInterceptTouchEvent(true);

自定义ViewPager：HorizontalScrollViewPager,然后将轮播图的Viewpager换成自定义的。

问题，此时在拖到轮播图的最后一个图片/第一个图片，再拖动应该响应外层的ViewPager。还有轮播图不应该拦截竖着拖动的事件，以免影响listview上下滑动。

```
public class HorizontalScrollViewPager extends ViewPager {
    private int downX;
    private int downY;
    public HorizontalScrollViewPager(Context context) {
        super(context);
    }
    public HorizontalScrollViewPager(Context context,
AttributeSet attrs) {
        super(context, attrs);
    }
    @Override
    public boolean dispatchTouchEvent(MotionEvent ev) {
        switch (ev.getAction()) {
            case MotionEvent.ACTION_DOWN:
```

// 按下的时候，需要设置不允许父元素拦截事件，只有自己能处理，才能获取到按下时的xy坐标

```
getParent().requestDisallowInterceptTouchEvent(true);
    downX = (int) ev.getX();
    downY = (int) ev.getY();
    break;
case MotionEvent.ACTION_MOVE:
    int moveX = (int) ev.getX();
    int moveY = (int) ev.getY();
    int diffX = moveX - downX;
    int diffY = moveY - downY;
    if(Math.abs(diffX) > Math.abs(diffY)) { //
当前是横向滑动，不让父元素拦截
        if(getCurrentItem() == 0 && diffX > 0) {
            // 显示的是第0张图，并且是从左向右滑动.

            getParent().requestDisallowInterceptTouchEvent(false);
        } else if(getCurrentItem() ==
(getAdapter().getCount() -1)
            && diffX < 0) {
            // 显示的是最后一张图，并且是从右向左滑动.

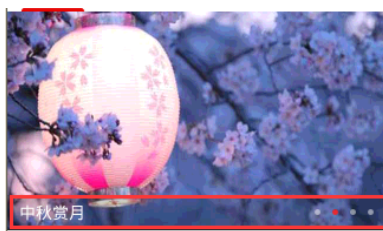
            getParent().requestDisallowInterceptTouchEvent(false);
        } else {

            getParent().requestDisallowInterceptTouchEvent(true);
        }
        } else { //当前是竖向滑动，让父元素拦截

            getParent().requestDisallowInterceptTouchEvent(false);
        }
        break;
    default:
        break;
    }
    return super.dispatchTouchEvent(ev);
}
```

## 轮播图的指示点

如图所示



## 1. 代码逻辑

以下代码写到NewsMenuTabDetailPager中的processData方法:

```
// 初始化轮播图对应的点
llPointGroup.removeAllViews(); // 把线性布局中所有的点清楚
View view;
LayoutParams params;
for (int i = 0; i < topNewsList.size(); i++) {
    view = new View(mContext);

    view.setBackgroundResource(R.drawable.tab_detail_topnews_point_bg);
    params = new LayoutParams(5, 5);
    if(i != 0) {
        params.leftMargin = 10;
    }
    view.setLayoutParams(params);
    view.setEnabled(false);
    llPointGroup.addView(view);
}

// 设置默认图片的描述和选中的点
previousEnabledPosition = 0;
TopNew topNew = topNewsList.get(previousEnabledPosition);
tvDescription.setText(topNew.title);
llPointGroup.getChildAt(previousEnabledPosition).setEnabled(true);
```

## 2. 点的选择器

在代码中给view设置的背景资源时，指定的是：tab\_detail\_topnews\_point\_bg

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_enabled="true"
android:drawable="@drawable/dot_focus" ></item>
    <item android:state_enabled="false"
android:drawable="@drawable/dot_normal" ></item>
</selector>
```

## 3. 设置默认选中的点

以下代码写到NewsMenuTabDetailPager中的processData方法:

```
private int previousEnabledPosition; // 前一个被选中的点的索引
```

```

        // 设置默认图片的描述和选中的点
        previousEnabledPosition = 0;
        TopNew topNew =
topNewsList.get(previousEnabledPosition);
        tvDescription.setText(topNew.title);

        llPointGroup.getChildAt(previousEnabledPosition).setEnabled(
true);

```

#### 4. 4viewpager切换，更换指示点，描述

给Viewpager添加pageChange监听：以下代码写到NewsMenuTabDetailPager中的processData方法，在实例化topNewViewPager时添加。

```
topNewViewPager.setOnPageChangeListener(this);
```

以下方法为viewpager的pagechange监听中的方法：

```

@Override
public void onPageSelected(int position) {
    // 把对应position的点给选中，并且把前一个被选中的点取消

    llPointGroup.getChildAt(previousEnabledPosition).setEnabled(
false);

    llPointGroup.getChildAt(position).setEnabled(true);
    previousEnabledPosition = position;

    tvDescription.setText(topNewsList.get(position).title);
}

```

### 轮播图实现轮播功能

实现原理：通过Handler发送延迟消息来实现。代码逻辑，以下代码写到NewsMenuTabDetailPager中的processData方法，加到方法末尾即可。

```

// 开始轮播图循环播放。
if(mHandler == null) {
    mHandler = new InternalHandler();
}
// 移除Handler对应消息队列中的回调和消息
mHandler.removeCallbacksAndMessages(null);
mHandler.postDelayed(new AutoSwitchPagerRunnable(),
5000); // 自动切换图片的子线程

```

➤ handler接收到消息后，怎么处理：

```

/**
 * @author andong
 * 内部消息处理器
 */
class InternalHandler extends Handler {

```

```

@Override
    public void handleMessage(Message msg) {
        // 切换图片.
        int currentItem =
(topNewViewPager.getCurrentItem() + 1) % topNewsList.size();
        topNewViewPager.setCurrentItem(currentItem);
        postDelayed(new AutoSwitchPagerRunnable(),
3000);
        //接收到消息后，隔3秒后再次执行自动切换，这样就可以实现轮播
    }
}

```

#### ➤ 自动切换图片的子线程

```

/**
 * @author wangdh
 * 自动切换图片的子线程
 */
class AutoSwitchPagerRunnable implements Runnable {
    @Override
    public void run() {
        // 得到一个消息发送给handler中的handleMessage方法.
        mHandler.obtainMessage().sendToTarget();
    }
}

```

### 手指对轮播图的控制

给轮播图片ImageView添加onTouchListener监听。手指按下时将handler中的消息移除就可以停止轮播，手指抬起时重新发送轮播消息。

如何判断图片是被点击的，因为图片是ViewPager中的item子项，所以图片会相应ViewPager的拖动切换操作，所以在这里如何区分点击和拖动？其实我们可以判断按下与抬起的时间间隔，如果小于500ms那么就说明是单击。代码如下：

```

/**
 * @author andong
 * 顶部轮播新闻中的图片的触摸事件
 */
class TopNewsItemTouchListener implements OnTouchListener {
    private int downX; //手指按下的x坐标
    private int downY; //手指按下的y坐标
    private long downTime; //按下时间

    @Override

```

```

        public boolean onTouch(View v, MotionEvent event) {
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    System.out.println("停止播放");

                    mHandler.removeCallbacksAndMessages(null); // 移除回调和消息停止
                    播放

                    downX = (int) event.getX();
                    downY = (int) event.getY();
                    downTime = System.currentTimeMillis();
                    break;
                case MotionEvent.ACTION_UP:
                    System.out.println("开始播放");
                    mHandler.postDelayed(new
                    AutoSwitchPagerRunnable(), 3000); // 重新开始
                    int upX = (int) event.getX();
                    int upY = (int) event.getY();

                    if (downX == upX && downY == upY) {
                        // 判断按下和抬起的时间是否超过了500毫秒.
                        long upTime =
                        System.currentTimeMillis();

                        long time = upTime - downTime;
                        if (time < 500) {
                            // 如何按下与抬起时间没有大于500ms, 那么就判定为点击事件
                            topNewItemClick(v);
                        }
                    }
                    break;
                default:
                    break;
            }
            return true;
        }

        public void topNewItemClick(View v) {
            System.out.println("轮播图的图片被点击了.");
        }
    }

```

## 图片三级缓存

- 内存缓存, 优先加载, 速度最快
- 本地缓存, 次优先加载, 速度快
- 网络缓存, 不优先加载, 速度慢, 浪费流量

## 内存溢出

- Android默认给每个app只分配16M的内存

- java中的引用

- 强引用 垃圾回收器不会回收, java默认引用都是强引用
- 软引用 `SoftReference` 在内存不够时,垃圾回收器会考虑回收
- 弱引用 `WeakReference` 在内存不够时,垃圾回收器会优先回收
- 虚引用 `PhantomReference` 在内存不够时,垃圾回收器最优先回收

注意: Android2.3+, 系统会优先将`SoftReference`的对象提前回收掉,即使内存够用

## LRUCache

least recently use 最少最近使用算法

会将内存控制在一定的大小内,超出最大值时会自动回收,  
这个最大值开发者自己定

## 新闻列表

新闻列表布局入下图所示



➤ 带边框图片的实现方式:

设置src指定将要显示的图片,然后设置padding=1,最后设置background背景色“@android:color/darker\_gray”布局的代码:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dip" >
```

```

<ImageView
    android:id="@+id/iv_tab_detail_news_item_image"
    android:layout_width="90dip"
    android:layout_height="65dip"
    android:background="@android:color/darker_gray"
    android:padding="1dip"
    android:scaleType="fitXY"
    android:src="@drawable/news_pic_default" />

<TextView
    android:id="@+id/tv_tab_detail_news_item_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dip"

    android:layout_toRightOf="@id/iv_tab_detail_news_item_image"
    android:lines="2"

    android:text="发牢骚肯德基发牢骚肯德基发牢骚款到即发阿萨德flak鸡丝豆腐"
    android:textSize="18sp" />

<TextView
    android:id="@+id/tv_tab_detail_news_item_time"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"

    android:layout_alignLeft="@id/tv_tab_detail_news_item_title"
    android:layout_below="@id/tv_tab_detail_news_item_title"
    android:layout_marginTop="5dip"
    android:text="1990-09-09 09:09:09"
    android:textColor="@android:color/darker_gray"
    android:textSize="14sp" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:src="@drawable/icon_news_comment_num" />
</RelativeLayout>

```

## 新闻列表的数据

### ➤ 数据适配器



NewsAdapter数据适配器：使用Holder、重用convertView优化。代码如下

:

```
class NewsAdapter extends BaseAdapter {
    @Override
    public int getCount() {
        return
newsList.size(); //newsList为数据集合，在下边给newsList赋值，这里可先不写
    }
    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        NewsViewHolder mHolder = null;
        if(convertView == null) {
            convertView = View.inflate(mContext,
R.layout.tab_detail_news_item, null);
            mHolder = new NewsViewHolder();
            mHolder.ivImage = (ImageView)
convertView.findViewById(R.id.iv_tab_detail_news_item_image);
            mHolder.tvTitle = (TextView)
convertView.findViewById(R.id.tv_tab_detail_news_item_title);
            mHolder.tvTime = (TextView)
convertView.findViewById(R.id.tv_tab_detail_news_item_time);
            convertView.setTag(mHolder); //
把mholder类设置convertView，为了下一次缓存时使用。
        } else {
            mHolder = (NewsViewHolder)
convertView.getTag();
        }
        // 把mHolder类中的对象赋值。
        NewsBean newsBean = newsList.get(position);
        bitmapUtils.display(mHolder.ivImage,
newsBean.listimage);
        mHolder.tvTitle.setText(newsBean.title);
        mHolder.tvTime.setText(newsBean.pubdate);
        return convertView;
    }
    @Override
    public Object getItem(int position) {
        return null;
    }
    @Override
    public long getItemId(int position) {
        return 0;
    }
}
```

```
}
```

### ➤ 数据初始化

以下代码添加到NewsMenuTabDetailPager.processData(String)方法

```
// 新闻列表数据初始化
newsList = bean.data.news;
if(newsAdapter == null) {
    newsAdapter = new NewsAdapter();
    newsListView.setAdapter(newsAdapter);
} else {
    newsAdapter.notifyDataSetChanged();
}
```

## 将轮播图加入到listview的顶部

1. 布局中将轮播图的相对布局移到新xml中: tab\_detail\_topnews.xml中

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="185dip" >
    <com.itheima41.zhbj.view.HorizontalScrollViewPager
        android:id="@+id/hsvp_tab_detail_topnews"
        android:layout_width="fill_parent"
        android:layout_height="185dip" />
    <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:layout_centerVertical="true"
        android:orientation="horizontal" >
        </LinearLayout>
    </RelativeLayout>
</RelativeLayout>

```

2. 代码中获取到新建的xml，反射成View，然后添加到listview的头中

```

View topNewsView = View.inflate(mContext,
R.layout.tab_detail_topnews, null);
ViewUtils.inject(this, topNewsView);
newsListView.addHeaderView(topNewsView);

```

**注意问题**，这时的轮播图无法显示。原因是我们在把轮播图添加到listview的头部的時候，这时候轮播图还没有初始化数据，如何解决？

将com.itheima41.zhbj.view.HorizontalScrollView的高度由fill\_parent改为与父元素一样的固定高度185dp。即使没有数据也给他撑开。

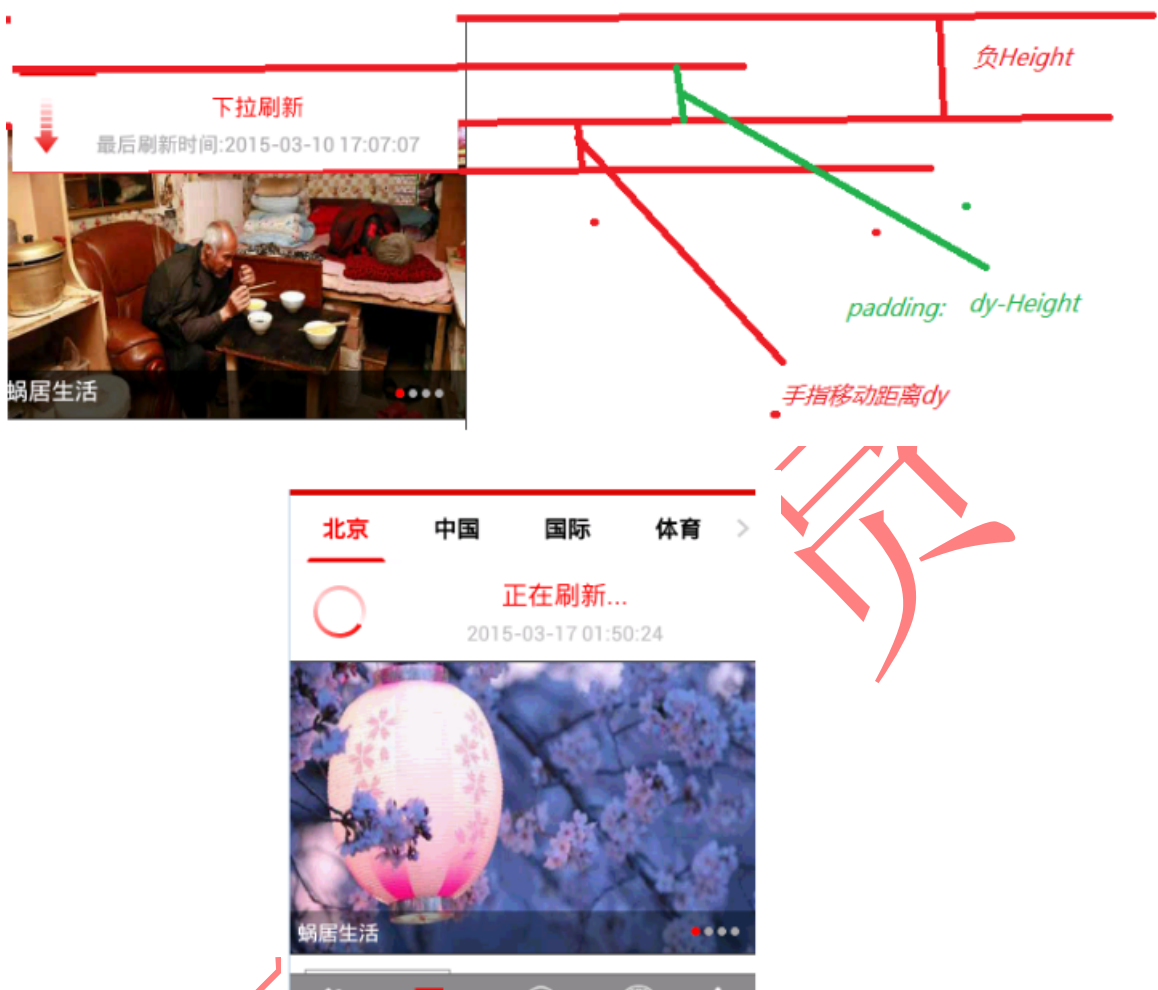
## 自定义下拉刷新、加载更多listview

- 下拉头布局refreshListView\_header.xml  
效果如下图所示：



下拉刷新的原理：

手指拖动布局，下滑，到达一定高度之后，状态由下拉刷新改变为松开刷新，如果继续下拉，则一直保持松开刷新状态，当松开后，控件会回弹到一定高度，变成正在刷新状态，获取到数据之后，或者超时之后，则刷新栏消失。两种状态由下面两个图所示：



通过对ListView添加了一个刷新layout（源代码res/layout/drop\_down\_to\_refresh\_list\_header.xml）作为header，在滚动中时不断改变header的高度和内容并记录一些状态，在用户手指离开屏幕时根据状态决定进行刷新还是放弃刷新。

主要是通过重写ListView的onTouchEvent和OnScrollListener的onScrollStateChanged、onScroll函数实现

先介绍下刷新状态共有四种，如下：

CLICK\_TO\_REFRESH 点击刷新状态，为初始状态；

DROP\_DOWN\_TO\_REFRESH 当刷新layout高度低于一定范围时，为此状态；

RELEASE\_TO\_REFRESH 当刷新layout高度高于一定范围时，为此状态；

REFRESHING 刷新中时，为此状态；

相信大家都知道这个是什么样子的效果。实现效果代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
```

```

<LinearLayout
    android:id="@+id/ll_refreshlistview_pull_down_header"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dip" >
        <ImageView
            android:id="@+id/iv_refreshlistview_header_arrow"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"

android:src="@drawable/common_listview_headview_red_arrow" />
        <ProgressBar
            android:id="@+id/pb_refreshlistview_header"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"

android:indeterminateDrawable="@drawable/custom_progressbar"
            android:visibility="invisible" />
        </FrameLayout>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:gravity="center_horizontal"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/tv_refreshlistview_header_state"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="下拉刷新"
            android:textColor="#FF0000"
            android:textSize="18sp" />
        <TextView

android:id="@+id/tv_refreshlistview_header_last_update_time"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dip"

```

```

        android:text="最后刷新时间： 1990-09-09 09:09:09"
        android:textColor="@android:color/darker_gray"
        android:textSize="14sp" />
    </LinearLayout>
</LinearLayout>
</LinearLayout>

```

#### ➤ 初始化下拉头：initPullDownHeaderView

问题：下拉刷新头，与轮播图头，谁在上谁在下。我们发现下拉刷新头是在自定义的RefreshListView的构造函数内初始化的（下拉刷新头属于RefreshListView，所以它是在内部完成初始化的），而轮播图是初始化后添加的（他是在NewsMenuTabDetailPager中通过listview.addHeaderView()添加的），所以下拉刷新在上。

初始化下拉头，具体代码如下所示：

```

/**
 * 初始化下拉头
 */
private void initPullDownHeaderView() {
    mHeaderView = (LinearLayout)
View.inflate(getContext(), R.layout.refreshlistview_header,
null);

    // 下拉刷新的头布局
    mPullDownHeaderView =
mHeaderView.findViewById(R.id.ll_refreshlistview_pull_down_header
);

    ivArrow = (ImageView)
mHeaderView.findViewById(R.id.iv_refreshlistview_header_arrow);
    mProgressBar = (ProgressBar)
mHeaderView.findViewById(R.id.pb_refreshlistview_header);
    tvState = (TextView)
mHeaderView.findViewById(R.id.tv_refreshlistview_header_state);
    tvLastUpdateTime = (TextView)
mHeaderView.findViewById(R.id.tv_refreshlistview_header_last_upd
ate_time);

    this.addHeaderView(mHeaderView);
}

```

#### ➤ 如何在RefreshListView获取轮播图的头布局

通过getChildAt(0)?这个是获取的整个头布局，包括下拉刷新。要想获取第二个头布局，可能需要通过getChildAt(0).xxx，所以这中方式不太好做。那么如何解决？

轮播图是属于我们自己添加的，所以我们可以自己提供一个方法，来添加这个轮播图，这时候就可以很方便的操作这个轮播图了。

```
/**
 * 添加一个自定义的头布局，加载下拉刷新的下面。
 * @param v
 */
public void addListViewCustomHeaderView(View v) {
}
```

定义完这个方法后，我们如何将自己添加的headerView添加到下拉刷新头布局下方？

其实我们可以在下拉刷新的布局外层添加一个线形布局，然后通过获取到这个跟布局，通过addView来实现。以下代码在addListViewCustomHeaderView方法中添加

```
mCustomHeaderView = v;
mHeaderView.addView(v); //mHeaderView就是下拉刷新布局的跟节点
```

#### ➤ 下拉刷新头默认隐藏

代码在initPullDownHeaderView中添加，如何实现？通过paddingTop来实现，如果设置的paddingTop值为负的下拉刷新布局的高度，就可以实现隐藏功能。

那么如何获取下拉刷新布局的高度？getHeight()这个是控件显示到界面上后调用，才会返回的高度。getmeaSureHeight()这个是控件经过测量后，才会返回的高度。刚初始化下拉刷新布局，还没显示，所以第一种不能用。但是也没有测量，不过我们可以手动测量。具体代码：

```
mPullDownHeaderView.measure(0, 0); // 自己测量自己的高度
mPullDownHeaderViewHeight =
mPullDownHeaderView.getMeasuredHeight();
System.out.println("下拉头布局的高度： " +
mPullDownHeaderViewHeight);
// 隐藏下拉头布局
mPullDownHeaderView.setPadding(0, -mPullDownHeaderViewHeight, 0,
0);
```

此代码要添加在initPullDownHeaderView方法的this.addHeaderView(mHeaderView);之前。

下拉滑动的时候，显示下拉刷新头。实现思路：

- 1、重写RefreshListView的onTouchEvent。
- 2、监听手势滑动，当listview显示的是顶部时，下拉出现下拉刷新头

如果显示的不是顶部，下拉则应该是显示相应的listview的条目。如何让下拉刷新头，跟着手指滑动，显示隐藏？实现思路：

1. 监听手指触摸事件，

2. 计算手指滑动距离，
3. 然后实时对下拉刷新头设置padding。

具体代码如下：

```
@Override
    public boolean onTouchEvent(MotionEvent ev) {
        switch (ev.getAction()) {
            case MotionEvent.ACTION_DOWN:
                downY = (int) ev.getY();
                break;
            case MotionEvent.ACTION_MOVE:
                if(downY == -1)
                { //有时候手指在快速按下时获取不到按下事件及坐标，所以需要在这里重新获取下
                    downY = (int) ev.getY();
                }
                int moveY = (int) ev.getY();
                int diffY = moveY - downY;

                if(diffY > 0 && getFirstVisiblePosition() == 0)
                { // 当前是向下滑动，并且是在ListView的顶部.
                    int paddingTop = -mPullDownHeaderViewHeight
+ diffY; //需要移动的大小
                    mPullDownHeaderView.setPadding(0,
paddingTop, 0, 0);
                    return true; // 自己来处理当前事件，
不响应父类的touch事件
                }
                break;
            case MotionEvent.ACTION_UP:
                break;
            default:
                break;
        }
        return super.onTouchEvent(ev);
    }
```

paddingTop是如何计算的，为什么等于-mPullDownHeaderViewHeight + diffY。因为下拉刷新也是通过设置padding让其隐藏的，setPadding(0, -mPullDownHeaderViewHeight, 0, 0)。所以要想让其跟着拖动的距离diffY来移动，就需要加上diffY。

#### ➤ 下拉刷新进度条样式





如图：左上角的圆圈。如何实现？肯定是一个ProgressBar，但是他的样式如何实现？其实可以通过shape。Shape如下：custom\_progressbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- rotate旋转
    fromDegrees开始角度
    pivotX, 旋转点, 50%自己的中心
-->
<rotate
xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="360" >

    <!-- innerRadiusRatio-
    >半径比: 控件宽度/这个比值, 得到的就是这个圆形的半径
        shape="ring"->形状: 圆形
        thicknessRatio-
    >厚度比: 控件宽度/这个比值, 得到的就是这个圆形线的厚度
        useLevel-
    >如果要使用LevelListDrawable对象, 就要设置为true。设置为true无渐变。false有渐变色
    -->
    <shape
        android:innerRadiusRatio="2.5"
        android:shape="ring"
        android:thicknessRatio="15"
        android:useLevel="false" >
        <!-- 渐变: 开始颜色, 中间颜色, 结束颜色
            type: linear 线性渐变, 这是默认设置
                radial 放射性渐变, 以开始色为中心。
                sweep 扫描线式的渐变。
        -->
```

```

        <gradient
            android:centerColor="#FF6666"
            android:endColor="#FF0000"
            android:startColor="#FFFFFF"
            android:type="sweep" />

    </shape>
</rotate>

```

➤ 然后给进度条设置样式

```

<ProgressBar
    android:id="@+id/pb_refreshlistview_header"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:indeterminateDrawable="@drawable/custom_progressbar"
    android:visibility="invisible" />

```

indeterminate不确定的意思。

➤ 下拉刷新状态的逻辑实现

即：向上箭头、向下箭头、进度条如何切换，何时切换？下拉刷新与松开刷新切换。

当下拉刷新头，完全显示时，显示松开刷新，也就是paddingTop=0，反之就是显示下拉刷新定义三种状态常量。

```

private final int PULL_DOWN_REFRESH = 0; // 下拉刷新状态
private final int RELEASE_REFRESH = 1; // 松开刷新状态
private final int REFRESHING = 2; // 正在刷新中状态

```

➤ 具体代码：onTouchEvent() 代码修改为：

```

if(diffY > 0 && getFirstVisiblePosition() == 0) { // 当前是向下滑动，
    并且是在ListView的顶部。
    int paddingTop = -mPullDownHeaderViewHeight +
diffY; //需要移动的大小
    //如果paddingTop<0时就显示下拉刷新并执行动画，但是如果在往上拉，paddingTop继续小于0，还会执行动画，我们这里只需要让他执行一次动画即可
    //所以当前只有不等于下拉刷新状态时，才能进入
    if(paddingTop < 0 && currentState != PULL_DOWN_REFRESH) {
        // 当前没有完全显示，并且当前状态属于松开刷新，进入下拉刷新
        System.out.println("下拉刷新");
        currentState =
PULL_DOWN_REFRESH; //并且进来以后立马让状态等于下拉刷新，避免重复进入
        refreshPullDownState();
    } else if(paddingTop > 0 && currentState != RELEASE_REFRESH)
    {

        // 当前完全显示，并且当前的状态是下拉刷新，进入到松开刷新
        System.out.println("松开刷新");
        currentState = RELEASE_REFRESH;
    }
}

```

```

        refreshPullDownState();
    }
    mPullDownHeaderView.setPadding(0, paddingTop, 0, 0);
    return true; // 自己来处理当前事件，不响应父类的touch事件
}

```

#### ➤ 箭头的动画

```

//在initPullDownHeaderView中调用此方法
private void initAnimation() {
    upAnimation = new RotateAnimation(0, -180, //
        Animation.RELATIVE_TO_SELF, 0.5f,
        Animation.RELATIVE_TO_SELF, 0.5f);
    upAnimation.setDuration(500);
    upAnimation.setFillAfter(true);

    downAnimation = new RotateAnimation(-180, -360,
        Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF,
        0.5f);
    downAnimation.setDuration(500);
    downAnimation.setFillAfter(true);
}

```

#### ➤ 根据状态来改变头布局

```

/**
 * 根据当前的状态currentState来刷新头布局
 */
private void refreshPullDownState() {
    switch (currentState) {
        case PULL_DOWN_REFRESH: // 下拉刷新
            // 箭头执行向下旋转的动画
            ivArrow.startAnimation(downAnimation);
            // 把状态修改为：下拉刷新
            tvState.setText("下拉刷新");
            break;
        case RELEASE_REFRESH: // 释放刷新
            // 箭头执行向下旋转的动画
            ivArrow.startAnimation(upAnimation);
            // 把状态修改为：下拉刷新
            tvState.setText("松开刷新");
            break;
        case REFRESHING: // 正在刷新中
            ivArrow.clearAnimation(); // 把动画清除掉
            ivArrow.setVisibility(View.INVISIBLE); // 隐藏箭头
            mProgressBar.setVisibility(View.VISIBLE);
            tvState.setText("正在刷新中..");
    }
}

```

```

        break;
    default:
        break;
    }
}

```

➤ 手指松开时，如何处理

以下代码在onTouchEvent中添加

```

case MotionEvent.ACTION_UP:
    downY = -1;
    if (currentState == PULL_DOWN_REFRESH) {
        // 当前是下拉刷新，把头布局隐藏
        mPullDownHeaderView.setPadding(0, -mPullDownHeaderViewHeight,
            0, 0);
    } else if (currentState == RELEASE_REFRESH) {
        // 当前是松开刷新，进入正在刷新中状态
        currentState = REFRESHING;
        refreshPullDownState();
        mPullDownHeaderView.setPadding(0, 0, 0,
            0); //这里要将头布局定位到左上角，因为会有"正在刷新中，没有位于左上角的情况"
    }
    break;

```

➤ 如果当前状态处于正在刷新的状态，就不能在下拉刷新在ACTION\_MOVE中添加此代码。在获取downY的值之后添加

```

// 如果当前状态是正在刷新中，直接跳出switch语句。
if (currentState == REFRESHING) {
    break;
}

```

➤ 如果当前显示的是轮播图，但是轮播图并未显示完全，这时用户再次下拉，应该响应listview的滚动事件，而不应该响应下拉刷新头的操作

在ACTION\_MOVE中添加：此代码添加在获取moveY值之前添加

```

// 如果轮播图的布局没有完全显示，不应该进行下拉头的操作，
// 而是响应ListView的本身的touch事件，直接跳出Switch语句。
if (mCustomHeaderView != null) {
    int[] location = new int[2]; // 第0位是x轴的地址，第1位是y轴的值
    if (mListViewOnScreenY == -1) {
        this.getLocationOnScreen(location); //
        获取ListView在屏幕中的坐标
        mListViewOnScreenY = location[1];
    }
    // 取出轮播图在屏幕中y轴的值
    mCustomHeaderView.getLocationOnScreen(location);
    // 如果轮播图在屏幕中y轴的值，小于 当前ListView在屏幕中y轴的值，

```

```
// 轮播图没有完全显示，不执行下拉头的操作，直接跳出。
if (location[1] < mListViewsOnScreenY) {
    // System.out.println("轮播图没有完全显示，直接跳出.");
    break;
}
}
```

➤ 如果用户手指抬起，并且触发了刷新操作，那么就要去获取数据如何实现？我们自定义的是一个公用组件，我们自己可以判断什么操作是刷新操作，但是刷新操作我们应该如何去调用用户自己定义的获取数据方法？这时，我们可以对外提供接口，使用回调。

先定义，监听事件：

```
/**
 * @author andong 自定义ListView刷新的监听事件
 */
public interface OnRefreshListener {
    /**
     * 当下拉刷新时，回调此方法。
     */
    public void onPullDownRefresh();
    /**
     * 当加载更多时触发此方法。此方法为加载更多时调用，我们提前写出来
     */
    public void onLoadingMore();
}
```

在定义接口，让用户传递实现后的监听事件。

```
/**
 * 设置ListView刷新事件的监听事件
 *
 * @param listener
 */
public void setOnRefreshListener(OnRefreshListener listener) {
    this.mOnRefreshListener = listener;
}
```

➤ 内部调用，监听方法

在ACTION\_UP中添加。注意要在状态为刷新中时调用

```
// 调用使用者的监听事件。
if (mOnRefreshListener != null) {
    mOnRefreshListener.onPullDownRefresh();
}
```

➤ 调用者，实现刷新数据接口，实现方法

在NewsMenuTabDetailPager中实现方法：

```
@Override
public void onPullDownRefresh() {
```

```

        HttpUtils utils = new HttpUtils();
        utils.send(HttpMethod.GET, url, new
RequestCallBack<String>() {
            @Override
            public void onSuccess(ResponseInfo<String>
responseInfo) {

                newsListView.OnRefreshDataFinish();//刷新完成回调
                Toast.makeText(mContext, "刷新数据成功",
0).show();

                CacheUtils.putString(mContext, url,
responseInfo.result); //缓存

                processData(responseInfo.result);
            }
            @Override
            public void onFailure(HttpException error,
String msg) {

                newsListView.OnRefreshDataFinish();
                Toast.makeText(mContext, "刷新数据失败",
0).show();
            }
        });
    }
}

```

➤ 刷新完成回调是干什么？

```

/**
 * 当用户刷新数据完成时，回调此方法，
把下拉刷新的头或者加载更多的脚给隐藏
 */
public void OnRefreshDataFinish() {
    // 当前是下拉刷新，隐藏头布局
    ivArrow.setVisibility(View.VISIBLE);
    mProgressBar.setVisibility(View.INVISIBLE);
    tvState.setText("下拉刷新");
    tvLastUpdateTime.setText("最后刷新时间：" +
getCurrentTime());
    mPullDownHeaderView.setPadding(0, -
mPullDownHeaderViewHeight, 0, 0);
    currentState = PULL_DOWN_REFRESH;
}

```

➤ 当前事件获取：

```
/**
```

```

    * 获取当前系统的时间，格式为：2014-11-16 16:07:12
    *
    * @return
    */
    public String getCurrentTime() {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        return format.format(new Date());
    }

```

➤ 加载更多

当用户滑动到底部的时候，我们需要获取更多的数据，效果如下图所示：



加载更多...

c

➤ 布局：refreshlistview\_footer.xml

横向线性布局，左侧一个菊花进度条，右侧一个TextView “加载更多……”

```

<?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="wrap_content"
    android:gravity="center"
    android:orientation="horizontal" >
    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dip"

        android:indeterminateDrawable="@drawable/custom_progressbar" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dip"
        android:text="加载更多..."

```

```

        android:textColor="#FF0000"
        android:textSize="25sp" />
</LinearLayout>

```

- 定义初始化加载更多脚的方法  
构造函数内调用，并且默认隐藏

```

/**
 * 初始化加载更多的脚布局
 */
private void initLoadMoreFooterView() {
    mFooterView = View.inflate(getContext(),
R.layout.refreshlistview_footer, null);
    mFooterView.measure(0, 0); // 测量自己
    mFooterViewHeight = mFooterView.getMeasuredHeight(); //
获取测量后的高度
    mFooterView.setPadding(0, -mFooterViewHeight, 0, 0); //
隐藏

    addFooterView(mFooterView);
    // 给当前ListView设置滚动的监听事件
    setOnScrollListener(this);
}

```

- ListView滑到底部时显示,添加滚动监听  
实现监听方法，如何监听滑动到底部？ListView实现OnScrollListener,并且  
重写方法

```

/**
 * 当滚动的状态改变时触发此方法. scrollState 当前的滚动状态
 *
 * SCROLL_STATE_IDLE 停止 SCROLL_STATE_TOUCH_SCROLL 触摸滚动
SCROLL_STATE_FLING
 * 快速的一滑
 */
@Override
public void onScrollStateChanged(AbsListView view, int scrollState) {
    // 当滚动停止时, 当前ListView是在底部(屏幕上最后一个显示的条目的索引是总长度 -
1)
    if (scrollState == SCROLL_STATE_IDLE || scrollState == SCROLL_STATE_FLING) { //
滚动停止
        System.out.println("滑动到底部了");
        // 显示脚布局
        mFooterView.setPadding(0, 0, 0, 0);
    }
}
/**
 * 当滚动时触发此方法
 */

```



```
@Override
public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount, int
totalCount) {
}
}
```

➤ 这时出现了一个问题，我们显示了脚布局，但是看不到，为什么？  
因为脚布局属于listview，我们显示脚布局时，listview展示的是最后一条item，我们及时通过setPadding()来让脚布局显示出来，但是listview并没有滚动，所以脚布局显示不出来。解决：只需添加如下代码即可。

```
// 让ListView滑动到底部，在显示脚布局后调用此方法
setSelection(getCount());
```

➤ 问题：当显示加载更多的脚布局时，再次拖动，还会进入显示脚布局的逻辑，如何解决？  
增加变量：isLoadingMore。是否正在加载更多中，默认false。然后增加条件：

```
if (scrollState == SCROLL_STATE_IDLE || scrollState ==
SCROLL_STATE_FLING) { // 滚动停止
    if ((getLastVisiblePosition() == getCount() - 1) &&
!isLoadingMore) {
        System.out.println("滑动到底部了");
        isLoadingMore = true; //进来后立马等于true，防止再次进入
    }
}
```

➤ 所有加载更多的逻辑已经写完，那么如何真正的请求数据去加载更多？  
其实跟下拉刷新是一个思路。对外暴露加载更多接口，OnRefreshListener中添加

```
/**
 * 当加载更多时触发此方法.
 */
public void onLoadingMore();
```

滑动监听内添加代码：onScrollStateChanged中添加

```
if(mOnRefreshListener != null) {
    mOnRefreshListener.onLoadingMore();
}
```

调用者：NewsMenuTabDetailPager实现onLoadingMore方法。more是加载更多的url，如果为空，则表明没有更多数据。加载更多我们这里就不缓存数据了。

```
@Override
public void onLoadingMore() {
    // 加载更多数据，去刷新更多的数据，并且把脚布局隐藏
    if (TextUtils.isEmpty(moreUrl)) {
        // 没有更多数据了。
    }
}
```

```

        Toast.makeText(mContext, "没有更多数据",
0).show();

        newsListView.OnRefreshDataFinish();
    } else {
        // 有更多的数据，去请求。
        HttpUtils utils = new HttpUtils();
        utils.send(HttpMethod.GET, moreUrl, new
RequestCallBack<String>() {
            @Override
            public void onSuccess(ResponseInfo<String>
responseInfo) {
                newsListView.OnRefreshDataFinish();
                Toast.makeText(mContext,
"加载更多数据成功", 0).show();
                TabDetailBean bean =
parserJson(responseInfo.result);
                // 把新闻列表数据取出来，
                并且在原有集合基础上加上。
                newsList.addAll(bean.data.news);
                newsAdapter.notifyDataSetChanged();
            }
            @Override
            public void onFailure(HttpException error,
String msg) {
                newsListView.OnRefreshDataFinish();
                Toast.makeText(mContext,
"加载更多数据失败", 0).show();
            }
        });
    }
}

```

➤ 那么加载更多数据完成后，应该如何做呢？

修改RefreshListView中的方法：

```

/**
 * 当用户刷新数据完成时，回调此方法，
把下拉刷新的头或者加载更多的脚给隐藏
 */
public void OnRefreshDataFinish() {
    if (isLoadingMore) {
        // 当前是加载更多，隐藏脚布局
        isLoadingMore = false;
        mFooterView.setPadding(0, -mFooterViewHeight, 0, 0);
    } else {
        // 当前是下拉刷新，隐藏头布局
    }
}

```

```
.....  
}  
}
```

- 给下拉刷新上拉加载加个开关  
默认是不开启的。

```
private boolean isEnabledPullDownRefresh = false; //  
是否启用下拉刷新的功能  
private boolean isEnabledLoadMoreRefresh = false; //  
是否启用加载更多的功能  
/**  
 * 设置是否启用下拉刷新  
 * @param b  
 */  
public void setEnabledPullDownRefresh(boolean b) {  
    isEnabledPullDownRefresh = b;  
}  
  
public void setEnabledLoadMoreRefresh(boolean b) {  
    isEnabledLoadMoreRefresh = b;  
}
```

具体条件在哪里添加：禁用下拉刷新功能在：onTouchEvent的Action\_Move里添加。

```
if(!isEnabledPullDownRefresh) { //未启用下拉刷新功能  
    break;  
}
```

禁用加载更多功能在：onScrollStateChanged中添加

```
if(!isEnabledLoadMoreRefresh) {  
    return;  
}
```

## 开源项目的提交和引用

- 提交开源引用

导入-现有工程从代码。

- 开源项目提交

1、首先需要有github的账号

2、新建资源库，选中Initializa this repository with a README(初始化一个readme文件)。这时你的资源库，默认有一个master分支。

本地使用git

clone，克隆刚创建的资源库（复制url，输入到克隆的url对话框即可）。

3、将提交的项目放到本地克隆的分支里，然后提交（提交时，会验证安装的git是否输入git的账号密码）。提交内容去掉：`.settings`、`bin`、`gen`、`.classpath`、`.project`。

4、然后提交（提交到本地），再push（push到服务器）。注意提交的项目名称，不要包含中文，不然github会出现中文乱码。

