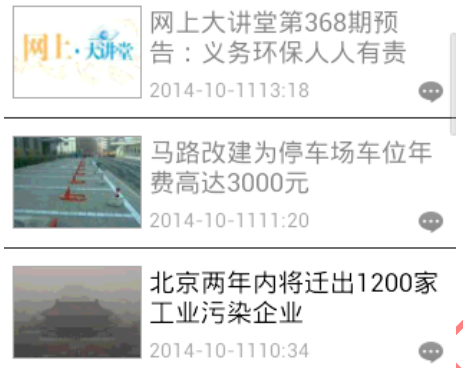
第4章新闻详情页

已读新闻

点击新闻进入新闻详情页后，返回列表，发现已读新闻的字体颜色变成灰色了，如图所示：



- 如何实现？

给listview添加item点击事件，记录查看过的新闻id，可以利用SP。根据sp取出的已读id，设置不同颜色。

- 刷新listview

在NewsAdapter中，从SP中取出已读新闻id，然后设置字体颜色

```
 // 取出已经缓存起来的id
String readIDArray = CacheUtils.getString(mContext,
readIDArrayKey, null);
if(!TextUtils.isEmpty(readIDArray) &&
readIDArray.contains(newsBean.id)) {
      // 当前缓存id不等于null，并且包含了当前新闻的id，就是已读新闻.
      mHolder.tvTitle.setTextColor(Color.GRAY);
} else {
      mHolder.tvTitle.setTextColor(Color.BLACK);
}
```

条目点击事件：需要实现OnItemClickListener

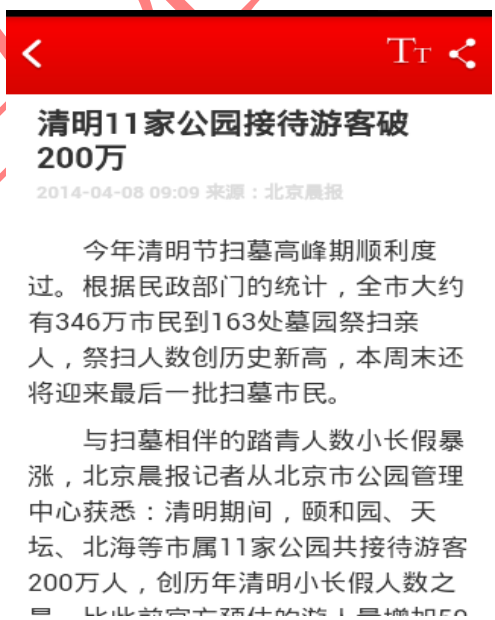
```
      private final String readIDArrayKey = "read_id_array";
@Override
      public void onItemClick(AdapterView<?> parent, View view,
int position,
                  long id) {
            NewsBean newsBean = newsList.get(position -
1);//这里需要减去1，因为这个position包含了listview的headerView。最好的做法是减去newsListView.getHeaderViewsCount();
            // 先取出以前存储的id，然后拼起来，再添加进去
```

```
            String readIDArray = CacheUtils.getString(mContext,
readIDArrayKey, null);

            if(!TextUtils.isEmpty(readIDArray))
{//如果已经存有id，那么就叠加
                  readIDArray = readIDArray + "," + newsBean.id;
            } else {//如果没有存有id，那么直接赋值
                  readIDArray = newsBean.id;
            }
            // 把当前查看的新闻的id存储起来.
            CacheUtils.putString(mContext, readIDArrayKey,
readIDArray);
            newsAdapter.notifyDataSetChanged();//刷新listview

            Intent intent = new Intent(mContext,
NewsDetailUI.class);
            intent.putExtra("url", newsBean.url);
            mContext.startActivity(intent);
      }
```

新闻详情页

新闻详情页通过webview来进行显示的。

- 界面的实现，如下图：



- 布局分析

顶部为导航栏，下边为webview或者是菊花进度条。所以外层是一个线形布局，内层webview与菊花可以放在帧布局里。

- 布局代码

```
<?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" >
   <include layout="@layout/title_bar" />
    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
        <WebView
            android:id="@+id/webview_news_detail"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
        <ProgressBar
            android:id="@+id/pb_news_detail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"

android:indeterminateDrawable="@drawable/custom_progressbar" />
    </FrameLayout>
</LinearLayout>
```

➢ WebView的使用

知道新闻详情页界面布局后，我们应该如何使用WebView加载网络界面。如何在Android中展示网络界面，mWebView.loadUrl(url);webview加载完url后，需要隐藏进度条。

```
            // 监听webview加载网页数据完成
            mWebView.setWebViewClient(new WebViewClient() {
                  /**
                   * 页面数据加载完成时回调
                   */
                  @Override
                  public void onPageFinished(WebView view, String
url) {
                        mProgressBar.setVisibility(View.GONE);
                  }
            });
```

➢ WebView默认不支持JS

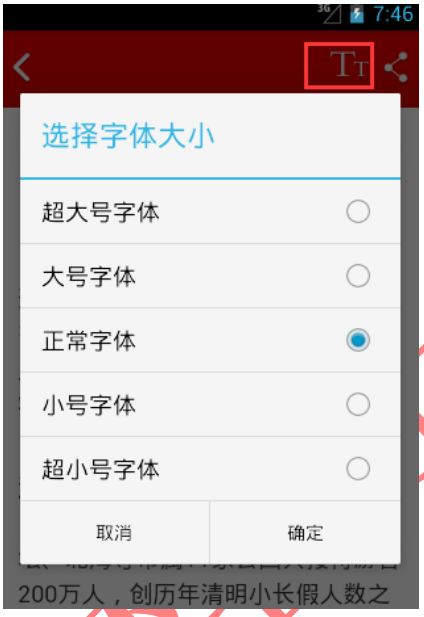
```
            settings =
mWebView.getSettings();//辅助类WebSettings，控制、设置webview
            settings.setJavaScriptEnabled(true); //
启用javascript的脚本功能
```

- WebView的缩放操作

```
            settings.setBuiltInZoomControls(true); //
屏幕上显示放大和缩小按钮
            settings.setUseWideViewPort(true);//
设置双击可以放大或者缩小
```

- 字体调整

点击右上角的字体调整按钮后出现，如何实现？



弹出选择字体大小的对话框

```
      private int currnetTextSizeIndex = 2; // 当前webview的字体
      /**
       * 弹出选择字体大小的对话框
       */
      private void showSelectTextSizeDialog() {
            Builder builder = new Builder(this);
            builder.setTitle("选择字体大小");
            String[] items = {"超大号字体", "大号字体", "正常字体",
"小号字体", "超小号字体"};
            builder.setSingleChoiceItems(items,
currnetTextSizeIndex, new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int
which) {
                        currnetTextSizeIndex = which;
                  }
            });
            builder.setPositiveButton("确定", new
DialogInterface.OnClickListener() {
                  @Override
```

```
                    public void onClick(DialogInterface dialog, int
which) {
                          switchTextSize();
                    }
              });
              builder.setNegativeButton("取消", null);
              builder.show();
        }
```

切换字体

```
        /**
         * 根据currentTextSizeIndex切换字体
         */
        protected void switchTextSize() {
              switch (currnetTextSizeIndex) {
              case 0:
                    settings.setTextSize(TextSize.LARGEST);
                    break;
              case 1:
                    settings.setTextSize(TextSize.LARGER);
                    break;
              case 2:
                    settings.setTextSize(TextSize.NORMAL);
                    break;
              case 3:
                    settings.setTextSize(TextSize.SMALLER);
                    break;
              case 4:
                    settings.setTextSize(TextSize.SMALLEST);
                    break;
              default:
                    break;
              }
        }
```

- 左上角返回按钮

直接关闭本界面即可。

- 右上角分享按钮

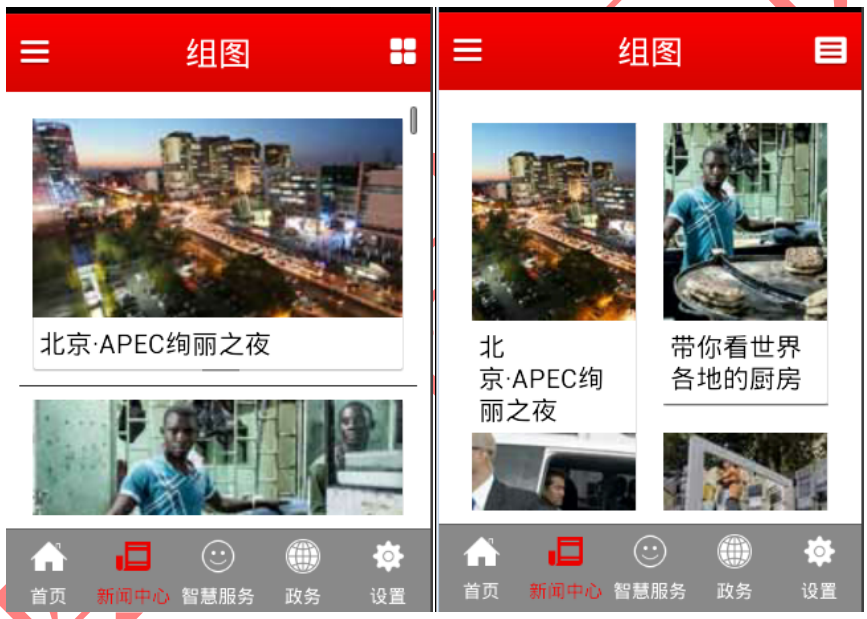
如果使用各个平台的sdk太麻烦，可以使用ShareSDK。ar文件在windows中如何运行javaw -jar xxx.jar

jdk安装版，在注册表内会有-jar这个参数，如果是绿色版就没有这个参数，就不能运行.jar文件。具体集成步骤：Android集成ShareSDK.htm具体网址：http://wiki.sharesdk.cn/Android_%E5%BF%AB%E9%80%9F%E9%9B%86%E6%88%90%E6%8C%87%E5%8D%97/

组图

点击新闻中心左侧菜单的“组图”

组图的展示形式如下图所示：



布局的实现

组图左上角有个切换按钮，点击后即可使新闻排列格式变化，如何实现。

- 第一种列表形式展示：listview
- 第二种表格形式展示：gridview- 2列，numColumns=2

具体代码：

```
<?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="match_parent"
    android:padding="10dip" >
```

```
    <ListView
        android:id="@+id/lv_photos_list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:cacheColorHint="@android:color/transparent" >
    </ListView>

    <GridView
        android:id="@+id/gv_photos_grid"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:cacheColorHint="@android:color/transparent"
        android:numColumns="2"
        android:visibility="gone" >
    </GridView>
</RelativeLayout>
```

页面显示方式的切换

当点击右上角按钮的时候么，实现不同图片显示的方式。其实实现的原理就是控制控件的显示和隐藏。具体代码如下：

1. 代码如下

```
    class OnPhotoListAndGridClickListener implements
OnClickListener {
            @Override
            public void onClick(View v) {
                PhotosMenuPager pager = (PhotosMenuPager)
v.getTag();
                pager.switchListOrGridPager((ImageButton) v);
//此方法是切换组图风格的
            }
    }
```

2. PhotosMenuPager添加切换方法

```
    /**
     * 切换当前页面
     */
    public void switchListOrGridPager(ImageButton ib) {
        if(isList) {
            // 切换到网格页面
            isList = false;
            mGridView.setVisibility(View.VISIBLE);
            mListView.setVisibility(View.GONE);
            mGridView.setAdapter(new PhotosAdapter());
```

```
        //如果是网格界面，那么图片就应该是列表图片，表示点击此图片就可以切换到列表页面

        ib.setImageResource(R.drawable.icon_pic_list_type);
            } else {
                // 切换到列表页面
                isList = true;
                mGridView.setVisibility(View.GONE);
                mListView.setVisibility(View.VISIBLE);
                mListView.setAdapter(new PhotosAdapter());


        ib.setImageResource(R.drawable.icon_pic_grid_type);
            }
        }
```

数据的获取及处理：请求数据代码：

```
@Override
        public void initData() {
            String json = CacheUtils.getString(mContext,
Constants.PHOTOS_URL, null);
            if(!TextUtils.isEmpty(json)) {
                processData(json);
            }
            HttpUtils httpUtils = new HttpUtils();
            httpUtils.send(HttpMethod.GET, Constants.PHOTOS_URL,
new RequestCallBack<String>() {
                @Override
                public void onSuccess(ResponseInfo<String>
responseInfo) {
                    System.out.println("组图数据请求成功: " +
responseInfo.result);
                    CacheUtils.putString(mContext,
Constants.PHOTOS_URL, responseInfo.result);
                    processData(responseInfo.result);
                }
                @Override
                public void onFailure(HttpException error,
String msg) {
                    System.out.println("组图数据请求失败: " +
msg);
                }
            });
        }
```

解析数据：

```
/**
        * 解析处理数据
        * @param result
        */
       protected void processData(String result) {
             Gson gson = new Gson();
             PhotosBean bean = gson.fromJson(result,
PhotosBean.class);
             photosList = bean.data.news;
             PhotosAdapter mAdapter = new PhotosAdapter();
             mListView.setAdapter(mAdapter);
       }
```

item布局的实现：

```
<?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:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dip"
        android:background="@drawable/pic_list_item_bg"
        android:fadingEdge="none"
        android:orientation="vertical" >
        <ImageView
            android:id="@+id/iv_photos_item_image"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:fadingEdge="none"
            android:minHeight="150dip"
            android:scaleType="centerCrop"
            android:src="@drawable/pic_item_list_default" />
        <TextView
            android:id="@+id/tv_photos_item_text"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dip"
            android:text="图片默认描述信息"
            android:textColor="#000000"
            android:textSize="20sp" />
    </LinearLayout>
```

```
</LinearLayout>
```

ImageView的scaleType，centerCrop是按比例扩大图片的size居中显示，使得图片长(宽)等于或大于view的长(宽)，会占满view的剩余空间。如果图片原本size就大于view，那么就只显示一部分。

- 适配器PhotosAdapter

代码如下：

```
class PhotosAdapter extends BaseAdapter {
            @Override
            public int getCount() {
                  return photosList.size();
            }
            @Override
            public View getView(int position, View convertView,
ViewGroup parent) {
                  PhotosViewHolder mHolder = null;
                  if(convertView == null) {
                        convertView = View.inflate(mContext,
R.layout.photos_item, null);
                        mHolder = new PhotosViewHolder();
                        mHolder.ivImage = (ImageView)
convertView.findViewById(R.id.iv_photos_item_image);
                        mHolder.tvText = (TextView)
convertView.findViewById(R.id.tv_photos_item_text);
                        convertView.setTag(mHolder);
                  } else {
                        mHolder = (PhotosViewHolder)
convertView.getTag();
                  }

                  PhotosItem photosItem =
photosList.get(position);
                  mHolder.tvText.setText(photosItem.title);
                  // 设置默认的图片，避免复用缓存时，图片错乱

      mHolder.ivImage.setImageResource(R.drawable.pic_item_list_de
fault);
                  // 给当前ivImage设置一个标识，为了方便在后期找到他
                  mHolder.ivImage.setTag(position);
                  // 请求网络，抓取图片.
                  Bitmap bm =
cacheUtils.getBitmapFromUrl(photosItem.listimage, position);
                  if(bm != null) {
                        mHolder.ivImage.setImageBitmap(bm);
                  }
```

```
                return convertView;
            }
            @Override
            public Object getItem(int position) {
                return null;
            }
            @Override
            public long getItemId(int position) {
                return 0;
            }
        }
        class PhotosViewHolder {
            public ImageView ivImage;
            public TextView tvText;

        }
```

图片的三级缓存

我们来新建一个图片工具类ImageCacheUtils.图片三级缓存，分为：

1. 从内存取.
2. 从本地取.
3. 从网络取.

代码如下：

```
/**
     * 根据url请求图片
     * 取图片的顺序:
     *                  1. 从内存取.
     *                  2. 从本地取.
     *                  3. 从网络取.
     * @param url
     * @param tag 当前请求的标识,图片所在listview的item的位置position
     * @return
     */
    public Bitmap getBitmapFromUrl(String url, int tag) {
        // 1. 从内存取.
        // 2. 从本地取.
        // 3. 从网络取.
        return null;
    }
```

- 从网络取图片
  1. 代码如下：

```
    /**
     * 根据url请求网络得到图片, 使用子线程执行.
```

```
     * @param url
     */
    private void getBitmapFromNet(String url, int tag) {
        new Thread(new RequestNetRunnable(url, tag)).start();
    }
```

2. 请求网络的任务类

```
    /**
     * @author andong
     * 请求网络的任务类
     */
    class RequestNetRunnable implements Runnable {

        private String url;
        private int tag; // 当前这次请求,
得到的图片需要设置给某个ImageView(身上有个tag和当前tag一样的ImageView)

        public RequestNetRunnable(String url, int tag) {
            this.url = url;
            this.tag = tag;
        }
        @Override
        public void run() {
            HttpURLConnection conn = null;
            try {
                URL mURL = new URL(url);
                conn = (HttpURLConnection)
mURL.openConnection();
                conn.setRequestMethod("GET");
                conn.setConnectTimeout(5000); //
设置连接超时时间为: 5秒钟
                conn.setReadTimeout(5000); //
设置读取超时时间为: 5秒钟。下载图片的时间
//                conn.connect();
                int responseCode =
conn.getResponseCode();//调用此方法是获取连接服务器后的响应码，此方法默认
会主动连接，所以可以不调用connect();
                if(responseCode == 200) {
                    InputStream is =
conn.getInputStream();
                    // 把流转换成Bitmap对象
                    Bitmap bm =
BitmapFactory.decodeStream(is);

                    // 把图片发送到主线程.
```

```
                                Message msg =
handler.obtainMessage();
                                msg.obj = bm;
                                msg.arg1 =
tag;//tag需要传递过去，处理消息时，要根据tag去找到相应的ImageView展示图片
                                msg.what = SUCCESS;
                                msg.sendToTarget(); //
把消息发送给PhotosMenuPager中消息处理器

                                // 向内存中存储一个.
                                mMemoryCache.put(url, bm);
//可以暂时不写

                                // 向本地中存储一个.
                                writeToLocal(bm, url);//可以暂时不写
                                return;
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        if(conn != null) {
                                conn.disconnect(); // 断开连接.
                        }
                }
                handler.obtainMessage(FAILED).sendToTarget();
        }
    }
```

3. 问题：

通过从网络获取此时图片可以正常显示，但是有个问题，图片显示的比较矮，高度小，如何解决？可以通过ImageView的minHeight来实现

➢ 从内存中获取

1. java中常用引用

强引用: 垃圾回收机制就程序崩溃都不会回收。软引用: 保证软件能够运行的情况, 可以回收软引用的对象。Map<String, SoftReference> map;SoftReference可以存图片

弱引用: 内存达到一定程度, 就会回收掉.虚引用: 只要垃圾回收一跑, 就会回收。google在3.0以后, 出现了一个LruCache集合, 用来缓存图片, 替换掉软引用.Lrucache:此类在android-support-v4的包中提供。

它的主要算法原理是把最近使用的对象用强引用存储在LinkedHashMap中，并且把最近最少使用的对象在缓存值达到预设定值之前从内存中移除。

在过去，我们经常会使用一种非常流行的内存缓存技术的实现，即软引用或弱引用 (SoftReference or WeakReference)。

但是现在已经不再推荐使用这种方式了，因为从 Android 2.3 (API Level 9)开始，垃圾回收器会更倾向于回收持有软引用或弱引用的对象，这让软引用和弱引用变得不再可靠。

- 使用Lrucache，具体实现

代码：

```
private LruCache<String, Bitmap> mMemoryCache;
//在构造函数中添加以下代码
// 获取模拟器运行时可以使用的内存大小 / 8，为什么除以8？
long maxMemory = Runtime.getRuntime().maxMemory() / 8;
//参数，Lrucache缓存大小,lrucache要计算缓存大小，需要知道每张图片的大小
mMemoryCache = new LruCache<String, Bitmap>((int) maxMemory) {
    //Bitmap即将缓存的图片
      @Override
      protected int sizeOf(String key, Bitmap value) {
            // 返回当前图片的大小，给Lrucache去计算当前缓存的内存大小

//getByteCount()//获取字节数量，就是图片的大小，但是这个方法要求api12以上
          //其实这个方法内部就是getRowBytes() * getHeight()
            return value.getRowBytes() * value.getHeight();
      }
};
```

拿到图片后，存带内存中，RequestNetRunnable中添加

```
// 向内存中存储一个.
mMemoryCache.put(url, bm);
```

在getBitmapFromUrl中添加代码

```
// 1. 从内存取.
Bitmap bm = mMemoryCache.get(url);
if (bm != null) {
    System.out.println("从内存中取");
    return bm;
 }
```

从本地取首先需要先存到本地。拿到图片后，存到本地，定义方法writeTo Local，RequestNetRunnable中调用

```
  private LruCache<String, Bitmap> mMemoryCache;
//构造中实例化
                    cacheDir = context.getCacheDir();
/**
     * 把图片缓存在本地
     *
     * @param bm
     * @param url
     */
    public void writeToLocal(Bitmap bm, String url) {
        try {
            String fileName = MD5Encoder.encode(url).substring(0,
10);//这里截取前十个字符当做是文件名
            FileOutputStream fos = new FileOutputStream(new
File(cacheDir, fileName));
            bm.compress(CompressFormat.JPEG, 100, fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

➢ MD5

```
import java.security.MessageDigest;

public class MD5Encoder {

      public static String encode(String string) throws Exception
{
          byte[] hash =
MessageDigest.getInstance("MD5").digest(string.getBytes("UTF-
8"));
          StringBuilder hex = new StringBuilder(hash.length * 2);
          for (byte b : hash) {
              if ((b & 0xFF) < 0x10) {
                  hex.append("0");
              }
              hex.append(Integer.toHexString(b & 0xFF));
          }
          return hex.toString();
      }
}
```

➢ 存到本地后，如何取

```
   /**
```

```
     * 根据url，从本地取图片
     *
     * @param url
     * @return
     */
    private Bitmap getBitmapFromLocal(String url) {
        try {
            Log.w("s", url);
            String fileName = MD5Encoder.encode(url).substring(0,
10);
            Log.w("s", fileName);
            File file = new File(cacheDir, fileName);
            if (file.exists()) {
                return BitmapFactory.decodeFile(file.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

➢ 调用：getBitmapFromUrl中添加以下代码

```
    // 2. 从本地取.
        bm = getBitmapFromLocal(url);
        if (bm != null) {
            System.out.println("从本地取");
            return bm;
        }
```

➢ 三级缓存的优化

我们每一次请求网络获取图片时，都是开启一个新的子线程去请求。如果可以通过线程池去管理这个请求网络的子线程，我们的程序性能会大大提升。具体做法：

```
private ExecutorService mExecutorService;
//实例化在构造中添加
// 获得一个固定线程为5个的线程池对象.
mExecutorService = Executors.newFixedThreadPool(5);
//修改原来的方法
private void getBitmapFromNet(String url, int tag) {
      // new Thread(new RequestNetRunnable(url, tag)).start();
      mExecutorService.execute(new RequestNetRunnable(url, tag));
  }
```