1.研究RecycleView的使用

2.RxJava再实践

RecyclerView对于整个应用都是必不可少的。

常见的 图片列表，文本列表，图文列表。

个性化一点的 其他View+RecyclerView的一些整体效果

发现轮子 compile 'com.jude:easyrecyclerview:4.0.6'

感谢开源。

3.嵌套html5页面（为图片添加点击事件）

##### RxJava使用封装

###### 原始

使用Rxjava 配合 Retrofit进行网络请求

服务器端返回的数据为

public class BaseResponse<T> {  
 public BaseResponseResult result = new BaseResponseResult();  
 public List<T> list;  
 public T model;  
}

如果服务器出现数据错误，则在result里面写入

public class BaseResponseResult {  
 public String requltType;  
 public String errorMessage;//提示信息  
  
 public boolean isOK(){  
 return MyStringUtils.*isEmpty*(errorMessage);  
 }  
}

另外支持列表和实体两种数据格式的返回。

正常情况下请求如下：

Model 只是为了使接口更加清晰

@Override  
public Observable<BaseResponse<News>> getNewsList(int picnews, int pageNum) {  
 return RetrofitSingleton.*apiService*.getNewsList(picnews,pageNum, C.*PAGE\_SIZE*);  
}

Presenter主要处理数据层逻辑

@Override  
public void refleshNewsList(int pictu) {  
 model.getNewsList(pictu,1)  
 .subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(new Observer<BaseResponse<News>>() {  
 @Override  
 public void onCompleted() {  
 pageNum = 2;  
 }  
  
 @Override  
 public void onError(Throwable e) {  
 e.printStackTrace();  
 view.setNewsPTListRefresh(null);  
 pageNum = 1;  
 }  
  
 @Override  
 public void onNext(BaseResponse<News> newsBaseResponse) {  
 for (News news : newsBaseResponse.list) {  
 if(news.picnews == 1)  
 news.picture = FrHereApi.*API\_URL* +news.picture;  
 }  
 view.setNewsPTListRefresh(newsBaseResponse.list);  
 }  
 });  
}

###### 问题

最终发现需要在每一个观察者的onNext方法中，判断数据是否正确，并且将网络请求配置为IO线程，再将UI操作切换至UI线程，这样就出现了较多的 “脏代码”。

###### 解决方法

public class RxHelper {  
 */\*\*  
 \* 对结果进行预处理 对返回值为List的结果进行处理  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* public static <T> Observable.Transformer<BaseResponse<T>, List<T>> handleResultList() {  
 return new Observable.Transformer<BaseResponse<T>, List<T>>() {  
 @Override  
 public Observable<List<T>> call(Observable<BaseResponse<T>> tObservable) {  
 return tObservable.flatMap(new Func1<BaseResponse<T>, Observable<List<T>>>() {  
 @Override  
 public Observable<List<T>> call(BaseResponse<T> result) {  
 if (result.result.isOK()) {  
 return *createData*(result.list);  
 } else {  
 return Observable.*error*(new NetworkErrorException(result.result.errorMessage));  
 }  
 }  
 }).subscribeOn(Schedulers.*io*()).observeOn(AndroidSchedulers.*mainThread*());  
 }  
 };  
 }  
  
 */\*\*  
 \* 对结果进行预处理 对返回值问model的结果进行处理  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* public static <T> Observable.Transformer<BaseResponse<T>, T> handleResultModel() {  
 return new Observable.Transformer<BaseResponse<T>, T>() {  
 @Override  
 public Observable<T> call(Observable<BaseResponse<T>> tObservable) {  
 return tObservable.flatMap(new Func1<BaseResponse<T>, Observable<T>>() {  
 @Override  
 public Observable<T> call(BaseResponse<T> result) {  
 if (result.result.isOK()) {  
 return *createData*(result.model);  
 } else {  
 return Observable.*error*(new NetworkErrorException(result.result.errorMessage));  
 }  
 }  
 }).subscribeOn(Schedulers.*io*()).observeOn(AndroidSchedulers.*mainThread*());  
 }  
 };  
 }  
 */\*\*  
 \* 创建成功的数据  
 \*  
 \** ***@param*** *data  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* private static <T> Observable<T> createData(final T data) {  
 return Observable.*create*(new Observable.OnSubscribe<T>() {  
 @Override  
 public void call(Subscriber<? super T> subscriber) {  
 try {  
 subscriber.onNext(data);  
 subscriber.onCompleted();  
 } catch (Exception e) {  
 subscriber.onError(e);  
 }  
 }  
 });  
 }  
}

*/\*\*  
 \* 针对网络请求封装的 观察者  
 \** ***@param*** <*T*>  
 *\*/*public abstract class RxSubscribe<T> extends Subscriber<T> {  
 @Override  
 public void onNext(T t) {  
 \_onNext(t);  
 }  
 @Override  
 public void onError(Throwable e) {  
 e.printStackTrace();  
 if (!NetUtils.*isNetworkConnected*(BaseApplication.*getInstance*())) {  
 \_onError("网络不可用");  
 } else if (e instanceof NetworkErrorException) {  
 \_onError(e.getMessage());  
 } else {  
 \_onError("请求失败，请稍后再试...");  
 }  
 }  
  
 protected abstract void \_onNext(T t);  
  
 protected abstract void \_onError(String message);  
  
}

###### 封装之后

然后使用的时候：

@Override  
public void refleshNewsList(int pic) {  
 model.getNewsList(pic,1)  
 .compose(ob)  
 .subscribe(new RxSubscribe<List<News>>() {  
 @Override  
 protected void \_onNext(List<News> newses) {  
 for (News news : newses) {  
 news.picture = FrHereApi.*API\_URL* +news.picture;  
 }  
 view.setNewsPTListRefresh(newses);  
 }  
  
 @Override  
 protected void \_onError(String message) {  
 view.showToast(message);  
 view.setNewsPTListRefresh(null);  
 pageNum = 1;  
 }  
  
 @Override  
 public void onCompleted() {  
 pageNum = 2;  
 }  
 });  
  
}

compose(ob) 方法处理了，上述的一些冗余操作。

##### **RecyclerView的研究**（easyRecycleView 库）

使用Recyclerview 代替 ListView

最吸引眼球的地方是

RecyclerArrayAdapter

BaseViewHolder

这两个类的封装

ViewHolder可以在多个Adapter里面使用，Adapter有封装了大量常用的方法。

Holder

*/\*\*  
 \* 新闻图片holder  
 \* Created by shli on 2016-08-11.  
 \*/*public class NewsPicViewHolder extends BaseViewHolder<News> {  
 private ImageView pic;  
 private TextView title;  
  
 public NewsPicViewHolder(ViewGroup parent) {  
 super(parent, R.layout.*fragment\_news\_pt\_pic\_item*);  
 title = $(R.id.*title*);  
 pic = $(R.id.*pic*);  
 }  
  
 @Override  
 public void setData(News data) {  
 title.setText(data.title);  
  
 Glide.*with*(getContext())  
 .load(data.picture)  
 .crossFade()  
 .diskCacheStrategy(DiskCacheStrategy.*ALL*)  
 .placeholder(R.drawable.*pic\_loading*)  
 .error(R.drawable.*pic\_loading*)  
 .into(pic);  
 }  
}

Adapter

*/\*\*  
 \* 图文新闻  
 \* Created by shli on 2016-08-11.  
 \*/*public class NewsPTAdapter extends RecyclerArrayAdapter<News> {  
 public NewsPTAdapter(Context context) {  
 super(context);  
 }  
  
 @Override  
 public int getViewType(int position) {  
 return getItem(position).picnews;  
 }  
  
 @Override  
 public BaseViewHolder OnCreateViewHolder(ViewGroup parent, int viewType) {  
 if (viewType == 1) {  
 return new NewsPicViewHolder(parent);  
 } else if (viewType == -1) {  
 return new NewsTextViewHolder(parent);  
 } else {  
 throw new InvalidParameterException();  
 }  
 }  
}

其实上面实现出来的 Adapter，是一个通用的Adapter，也就是说就算不是使用这个库，Adapter也是可以给原生的RecyclerView使用的。

并且adapter支持数据管理，Header与Footer添加，加载更多。没有更多。加载错误

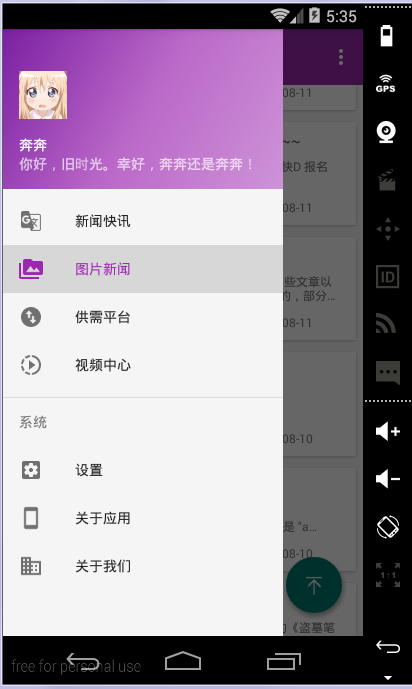
<com.jude.easyrecyclerview.EasyRecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_progress="@layout/custom\_loading\_view"  
 app:layout\_error="@layout/custom\_error\_view"  
 app:layout\_empty="@layout/custom\_empty\_view"  
 />

Xml文件支持配置 各种状态页

具体的使用方法参见：

<https://github.com/Jude95/EasyRecyclerView/blob/master/README_ch.md>

###### 使用效果：



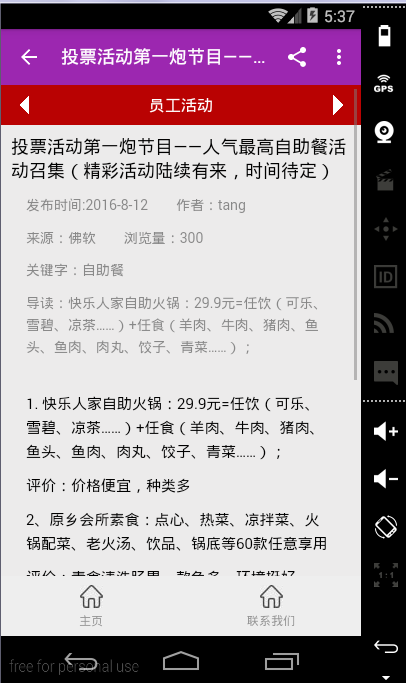


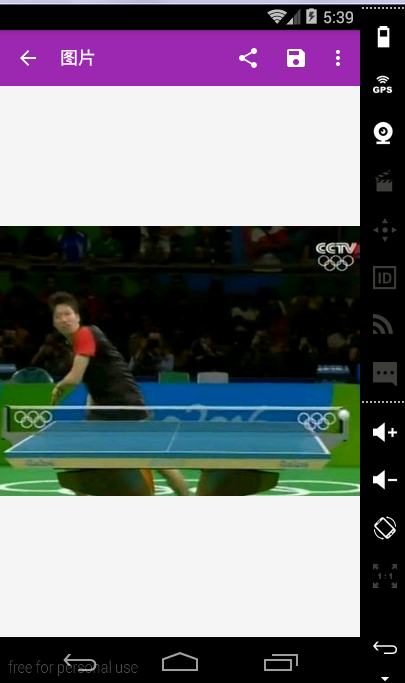






##### 详情页面 嵌套Html5 页面





难点主要在一些细节的控制上面

1. 对加载动画的控制

最初是这样实现的，但是测试过程中发现，正在加载的时候关闭页面会导致app crash。

/\*webView.setWebChromeClient(new WebChromeClient() {  
 @Override  
 public void onProgressChanged(WebView view, int newProgress) {  
 super.onProgressChanged(view, newProgress);  
 if (newProgress == 100) {  
 multiplestatusview.setVisibility(View.GONE);  
 }  
 }  
});\*/

解决方法：在页面Finish里面回调目前没有发现问题

webView.setWebViewClient(new WebViewClient() {  
 @Override  
 public void onPageFinished(WebView view, String url) {  
 super.onPageFinished(view, url);  
 *addImageClickListner*(webView);  
 multiplestatusview.setVisibility(View.*GONE*);  
 }  
});

1. 为图片添加点击事件，点击进入图片浏览页面，并且支持保存和放大。

稍微封装一下，其实并不好，因为覆盖了系统的回调方法。

正式的系统计划另外在提供三个回调给开发者，这样就基本没问题了

// 注入js函数监听  
private static void addImageClickListner(WebView webView) {  
 // 这段js函数的功能就是，遍历所有的img几点，并添加onclick函数，函数的功能是在图片点击的时候调用本地java接口并传递url过去  
 //onClick=\"javascript:mWebViewImageListener.onImageClick('$2')\"")  
 webView.loadUrl("javascript:(function(){" +  
 "var objs = document.getElementsByTagName(\"img\"); " +  
 "for(var i=0;i<objs.length;i++) " +  
 "{"  
 + " objs[i].onclick=function() " +  
 " { "  
 + " WebViewImageListener.onImageClick(this.src); " +  
 " } " +  
 "}" +  
 "})()");  
}

*/\*\*  
 \* 为图片添加点击事件  
 \** ***@param*** *webView 当前web容器  
 \** ***@param*** *onWebViewImageListener 点击事件的回调<br> <font color="red">**一定要在方法上添加@JavascriptInterface注解 </br></font>  
 \*/*@SuppressLint({ "SetJavaScriptEnabled", "JavascriptInterface" })  
public static void addOnclickToHtml(final View multiplestatusview,final WebView webView ,OnWebViewImageListener onWebViewImageListener){  
 webView.addJavascriptInterface(onWebViewImageListener, "WebViewImageListener");  
 webView.setWebViewClient(new WebViewClient() {  
 @Override  
 public boolean shouldOverrideUrlLoading(WebView view, String url) {  
 view.loadUrl(url);  
 return true;  
 }  
 @Override  
 public void onPageStarted(WebView view, String url, Bitmap favicon) {  
 super.onPageStarted(view, url, favicon);  
 }  
 @Override  
 public void onPageFinished(WebView view, String url) {  
 super.onPageFinished(view, url);  
 *addImageClickListner*(webView);  
 multiplestatusview.setVisibility(View.*GONE*);  
 }  
 });  
}

###### 使用

ImgHelper.*addOnclickToHtml*(multiplestatusview, webView, new OnWebViewImageListener() {  
 @Override  
 @JavascriptInterface  
 public void onImageClick(String url) {  
 start(ImageFragment.*newInstance*(url));  
 }  
});

@JavascriptInterface这个注解必须要加上否则找不到方法。

并且注意对应生命周期

@Override  
public void onResume() {  
 super.onResume();  
 if (webView != null){  
 webView.onResume();  
 }  
}  
  
@Override  
public void onPause() {  
 super.onPause();  
 if (webView != null){  
 webView.onPause();  
 }  
}  
  
@Override  
public void onStop() {  
 super.onStop();  
 if (webView != null){  
 webView.stopLoading();  
 }  
}

@Override  
public void onDestroyView() {  
 super.onDestroyView();  
 if(webView !=null){  
 webView.removeAllViews();  
 webView.destroy();  
 }  
}

#### 总结

MultipleStatusView

这个类封装了各种页面状态，使用起来方便。

另外：

在app的启动Activity页面配置

android:launchMode="singleTask"

会出现app每次启动都会进入启动页面的Activity。