# 我的商城主线笔记

### 启动页

#### 去标题栏

**public class** WelcomeActivity **extends** Activity

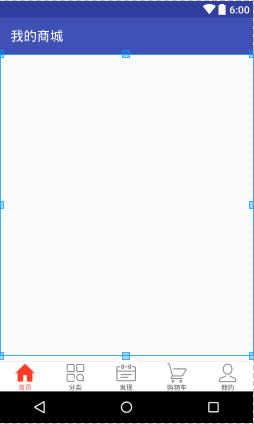
#### 延时3s进入主页面中的延时实现

**private** Handler **handler** = **new** Handler();

**handler**.postDelayed(**new** Runnable() **{** @Override  
 **public void** run() **{** // 进到主页面  
 finish(); *// 关闭本页面* **}  
}**, 3000);

### MainActivity主页

#### 主页布局（整体：线性垂直布局）



RadioGroup水平布局（带背景）

帧布局

##### RadioButton代码

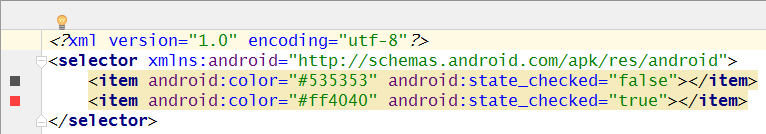
// 首页RadioButton

<**RadioButton** android:id="@+id/rb\_home"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/home\_button\_selector"  
 android:text="@string/rb\_home"  
 android:checked="true"/>

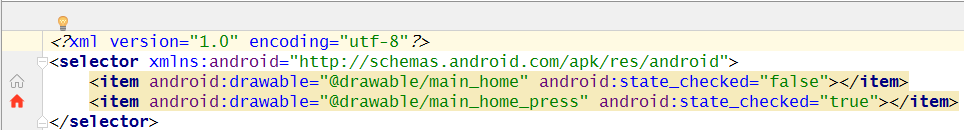
##### RadioButton样式

<**style** name="MainButtonStyle">  
 *<!-- Customize your theme here. -->* <**item** name="android:layout\_width">0dp</**item**>  
 <**item** name="android:layout\_height">wrap\_content</**item**>  
 <**item** name="android:layout\_weight">1</**item**>  
 <**item** name="android:button">@null</**item**>  
 *<!-- <item name="android:drawablePadding">3dp</item>-->* <**item** name="android:textColor">@drawable/bottom\_button\_text\_selector</**item**> // 文本选择器  
 <**item** name="android:textSize">10sp</**item**>  
 <**item** name="android:gravity">center</**item**>  
</**style**>

bottom\_button\_text\_selector文本选择器

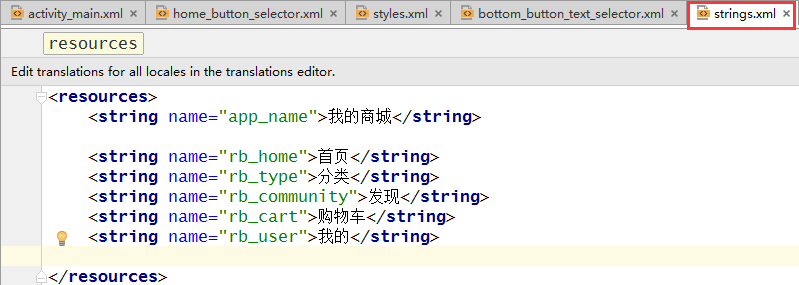


##### RadioButton顶部按钮选择器



##### RadioButton名称管理

android:text="@string/rb\_home"



##### 完整代码（注意中间的RadioButton）

*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.myemcu.myshop.app.MainActivity">  
  
 <**FrameLayout** android:id="@+id/frame\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"/>  
  
 <**RadioGroup** android:id="@+id/rg\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/home\_bottom\_parent\_bg"  
 android:orientation="horizontal">  
  
 <**RadioButton** android:id="@+id/rb\_home"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/home\_button\_selector"  
 android:text="@string/rb\_home"  
 android:checked="true"/>  
  
 <**RadioButton** android:id="@+id/rb\_type"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/type\_button\_selector"  
 android:text="@string/rb\_type"/>  
  
 <**RadioButton** android:id="@+id/rb\_community"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/community\_button\_selector"  
 android:paddingTop="10dp"  
 android:text="@string/rb\_community"/>  
  
 <**RadioButton** android:id="@+id/rb\_cart"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/cart\_button\_selector"  
 android:text="@string/rb\_cart"/>  
  
 <**RadioButton** android:id="@+id/rb\_user"  
 style="@style/MainButtonStyle"  
 android:drawableTop="@drawable/user\_button\_selector"  
 android:text="@string/rb\_user"/>  
  
 </**RadioGroup**>  
  
</**LinearLayout**>

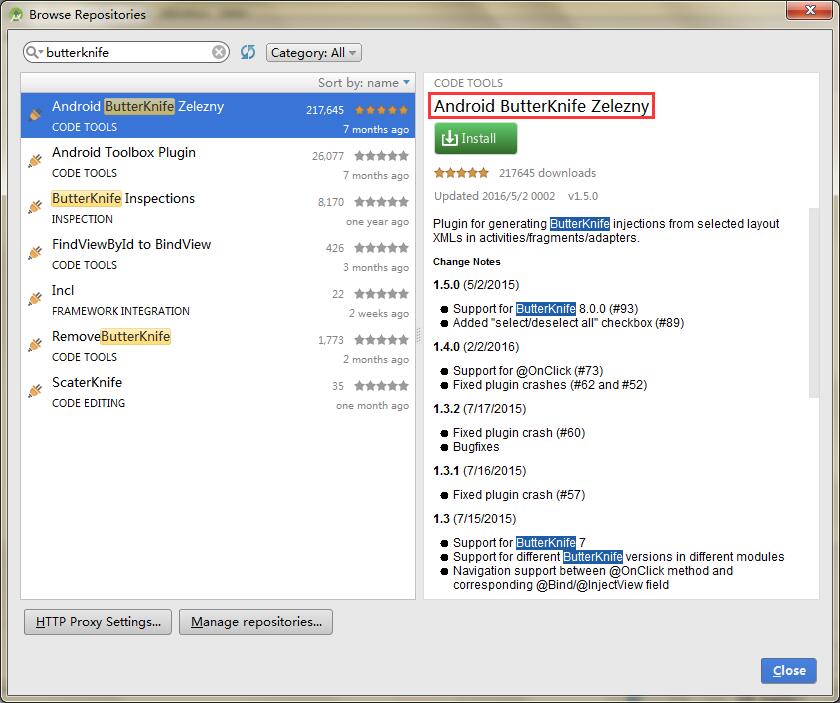
#### 使用Butterknife实例化主页控件

##### 添加依赖（须联网）

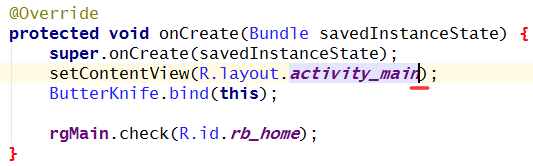
compile 'com.jakewharton:butterknife:7.0.1' // 稳定版，8.4.0的不稳定

##### 下载插件

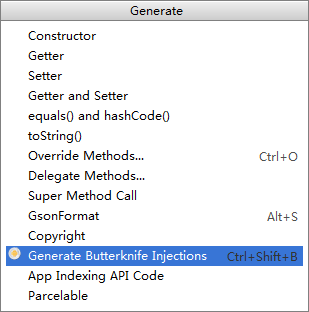
如下图所示：



##### 设置光标位置



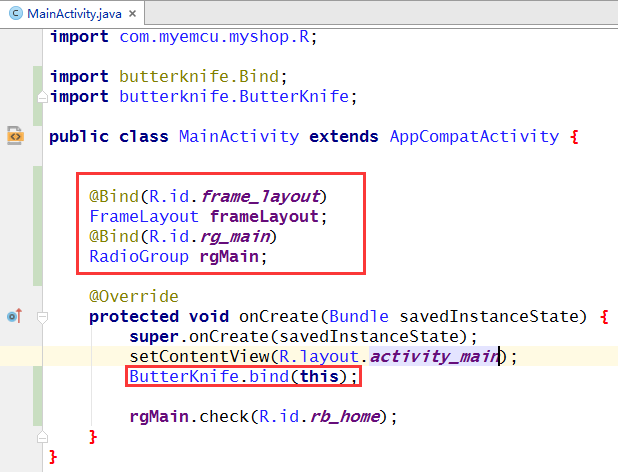
##### Alt+Insert



##### 选择需要被实例化的控件



##### 实现效果



### 系统框架——软件分包

#### 功能分包

* app——Activity
* base——公共基类
* home
* type
* community
* cart
* user

——上面5个，子Fragment。

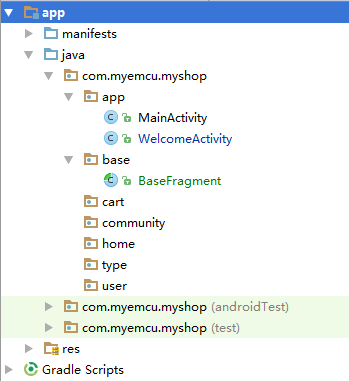
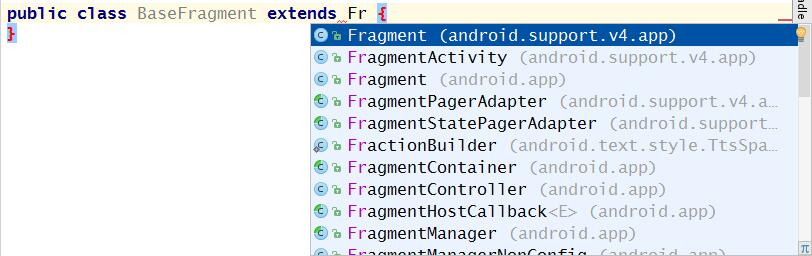


图 1.3.1 软件分包

#### 5个Fragment页面的公共Fragment

##### 继承时用V4包



##### BaseFragment

**public abstract class** BaseFragment **extends** Fragment **{  
  
 protected** Context **context**; *// (public全公开；privite不公开，仅自己用；protected公开到熟人)* @Override **public void onCreate**(@Nullable Bundle savedInstanceState) **{** *// 该类被系统创建时回调* **super**.onCreate(savedInstanceState);  
 **context**=getActivity();  
 **}** @Nullable  
 @Override  
 **public** View **onCreateView**(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle

savedInstanceState) **{  
 return** initView();  
 **}**

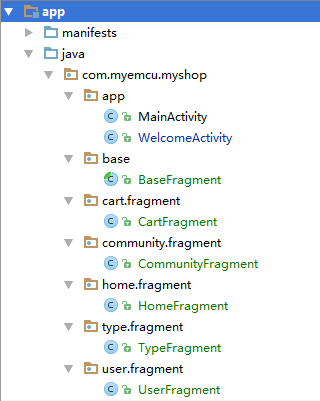
**public abstract** View initView(); *// 由继承它的子Fragment强制实现，当然基类也必须是抽象的* @Override *// 当Activity被创建的时候，回调该方法* **public void onActivityCreated**(@Nullable Bundle savedInstanceState) **{  
 super**.onActivityCreated(savedInstanceState);  
 initData();  
 **}****public void** initData() **{** *// 当子Fragment需要联网请求数据的时候，可重写该方法* **}  
}**

### 简单构建5个Fragment页面内容

#### 以如下HomeFragment为模板

*// 首页Fragment***public class** HomeFragment **extends** BaseFragment**{  
  
 private** TextView **textView**;  
  
 @Override  
 **public** View **initView**() **{  
 textView**=**new** TextView(**context**);  
  
 Log.e("TAG","首页Fragment-UI-已初始化...");  
  
 **textView**.setGravity(Gravity.***CENTER***);  
 **textView**.setTextSize(25);  
 **textView**.setTextColor(Color.***BLUE***);  
 **return textView**;  
 **}** @Override  
 **public void initData**() **{  
 super**.initData();  
  
 Log.e("TAG","首页Fragment-数据-已初始化...");  
  
 **textView**.setText("我是首页...");  
 **}  
}**

#### 完善其它Fragment（卡卡西模式）



### 在MainActivity中，监听切换5个Fragment（UI框架完）

#### 添加Fragment

**private** ArrayList<BaseFragment> **fragments**;

**fragments** = **new** ArrayList<>(); *// 新建集合对象***fragments**.add(**new** HomeFragment()); *// 添加 首页 Fragment***fragments**.add(**new** TypeFragment()); *// 添加 分类 Fragment***fragments**.add(**new** CommunityFragment()); *// 添加 社区 Fragment***fragments**.add(**new** CartFragment()); *// 添加 购物车 Fragment***fragments**.add(**new** UserFragment()); *// 添加 用户 Fragment*

#### 监听RadioGroup

**rgMain**.**setOnCheckedChangeListener**(**new** RadioGroup.OnCheckedChangeListener() **{** @Override  
 **public void** onCheckedChanged(RadioGroup group, **int** checkIid) **{** **}  
}**);

#### 取得当前位置的Fragment

**private** BaseFragment getFragment(**int** postion) **{**

**}**

#### 切换到需要显示的Fragment

**private void** switchFragment(Fragment preFragment, BaseFragment currentFragment) **{  
  
}**

#### 完整代码

**public class** MainActivity **extends FragmentActivity** **{** // 1 去标题栏 2 支持getSupportFragmentManager() **private** ArrayList<BaseFragment> **fragments**; **private int postion** = 0;  
  
 **private** Fragment **tempFragment**; *// 临时Fragment(缓存Fragment)* **private** BaseFragment **currentFragment**; *// 当前Fragment* @Bind(R.id.***frame\_layout***)  
 FrameLayout **frameLayout**;  
 @Bind(R.id.***rg\_main***)  
 RadioGroup **rgMain**;  
  
 @Override  
 **protected void onCreate**(Bundle savedInstanceState) **{  
 super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 ButterKnife.bind(**this**); *// 自动生成* initFragment(); *// 初始化所有Fragment* initListener(); *// 初始化页面控件监听* **}** *// 5个Fragment页面(按顺序添加)* **private void** initFragment() **{  
 fragments** = **new** ArrayList<>(); *// 新建集合对象* **fragments**.add(**new** HomeFragment()); *// 添加 首页 Fragment* **fragments**.add(**new** TypeFragment()); *// 添加 分类 Fragment* **fragments**.add(**new** CommunityFragment()); *// 添加 社区 Fragment* **fragments**.add(**new** CartFragment()); *// 添加 购物车 Fragment* **fragments**.add(**new** UserFragment()); *// 添加 用户 Fragment* **}  
  
 private void** initListener() **{** *// 首页中的各种监听* **rgMain**.setOnCheckedChangeListener(**new** RadioGroup.OnCheckedChangeListener() **{** @Override  
 **public void** onCheckedChanged(RadioGroup group, **int** checkIid) **{  
 switch** (checkIid) **{  
 case** R.id.***rb\_home***: **postion** = 0;  
 **break**;  
  
 **case** R.id.***rb\_type***: **postion** = 1;  
 **break**;  
  
 **case** R.id.***rb\_community***: **postion** = 2;  
 **break**;  
  
 **case** R.id.***rb\_cart***: **postion** = 3;  
 **break**;  
  
 **case** R.id.***rb\_user***: **postion** = 4;  
 **break**;  
  
 **default**: **postion** = 0; **break**;  
 **}** *// 根据位置取当前Fragment* **currentFragment** = getFragment(**postion**);  
 *// 切换Fragment* switchFragment(**tempFragment**, **currentFragment**); *// 临时Fragment，当前取出的Fragment* **}  
 }**);  
  
 **rgMain**.check(R.id.***rb\_home***); *// 默认选中首页(用8.4.0的Butterknife这里要崩)* **}** *// 根据位置，得到相应Fragment* **private** BaseFragment getFragment(**int** postion) **{  
  
 if** (**fragments** != **null** && **fragments**.size()>0) **{** BaseFragment baseFragment = **fragments**.get(postion);  
 **return** baseFragment;  
 **}else {  
 return null**;  
 **}  
 }** *// 根据上次preFragment与当前的Fragment进行切换* **private void** switchFragment(Fragment preFragment, BaseFragment currentFragment) **{**

**if** (**tempFragment** != currentFragment) **{** *// Fragment一样，就不需要处理了。* **tempFragment** = currentFragment;

**if** (currentFragment != **null**) **{**

FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
 **if** (!currentFragment.isAdded()) **{  
 if** (preFragment != **null**) **{** transaction.hide(preFragment);  
 **}** transaction.add(R.id.***frame\_layout***,currentFragment).commit();  
 **}else {  
 if** (preFragment != **null**) **{** transaction.hide(preFragment);  
 **}** transaction.show(currentFragment).commit();  
 **}  
 }** *// 验证同一个地方只加载一次Fragment  
 // Toast.makeText(MainActivity.this,"我进来了",Toast.LENGTH\_SHORT).show();* **}  
 }  
}**

#### 实现效果



### 实现首页HomeFragment布局

#### 布局方案（整体相对布局）

* Titlebar
* RecyclerView
* ImageView

#### Titlebar

##### 实现效果



##### 实现代码

*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="horizontal"  
 android:background="#ed3f3f"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <**TextView** android:id="@+id/tv\_search\_home"  
 android:layout\_width="0dp"  
 android:layout\_weight="1"  
 android:layout\_height="35dp"  
 android:layout\_marginLeft="10dp"  
 android:text="输入搜索信息"  
 android:textSize="13sp"  
 android:drawablePadding="10dp"  
 android:padding="5dp"  
 android:background="@drawable/search\_home\_shape"  
 android:drawableLeft="@drawable/home\_search\_icon" // 放大镜图标  
 android:layout\_gravity="center"  
 android:gravity="center\_vertical"  
 tools:ignore="HardcodedText,RtlHardcoded"/>  
  
 <**TextView** android:id="@+id/tv\_message\_home"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:text="消息"  
 android:textColor="#fff"  
 android:drawableTop="@drawable/new\_message\_icon" // 信封图标  
 tools:ignore="HardcodedText"/>  
  
</**LinearLayout**>

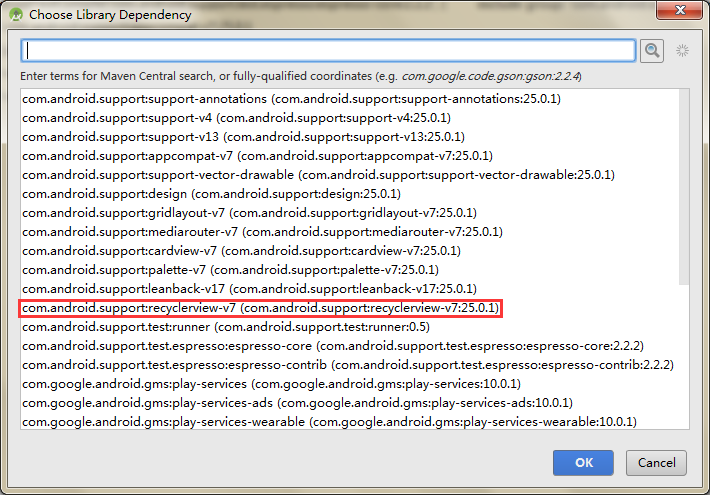
其中，前一个TextView的背景代码如下

*<?*xml version="1.0" encoding="utf-8"*?>*<**shape** xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <**solid** android:color="#fff" />  
 <**corners** android:radius="20dp" />  
</**shape**>

#### RecyclerView

<**android.support.v7.widget.RecyclerView** android:id="@+id/rv\_home"  
 android:layout\_below="@+id/titlebar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>

说明：必须先添加依赖（版本要与com.android.support:appcompat-v7:25.0.1一致）



#### ImageView

实现代码

<**ImageView** android:id="@+id/ib\_top"  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:background="@drawable/top\_btn"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginBottom="20dp"  
 android:layout\_marginRight="20dp"  
 android:visibility="visible"  
 tools:ignore="ContentDescription,RtlHardcoded"/>

说明：图片必须是透明背景的.png格式，本图大小256x256

#### 整体实现

##### 实现代码

*<?*xml version="1.0" encoding="utf-8"*?>*<**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">  
  
 <**include** android:id="@+id/titlebar"  
 layout="@layout/titlebar"/>  
 <**android.support.v7.widget.RecyclerView** android:id="@+id/rv\_home"  
 android:layout\_below="@+id/titlebar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
  
 <**ImageView** android:id="@+id/ib\_top"  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:background="@drawable/top\_btn"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginBottom="20dp"  
 android:layout\_marginRight="20dp"  
 android:visibility="visible"  
 tools:ignore="ContentDescription,RtlHardcoded"/>  
  
</**RelativeLayout**>

##### 实现效果



### 初始化首页HomeFragment布局

#### 实例化RecyclerView、ImageView、TextView（两个）

#### 监听ImageView、TextView（两个）

#### 关键代码

**recy\_home** = (RecyclerView) view.findViewById(R.id.***recy\_home***);  
**iv\_top** = (ImageView) view.findViewById(R.id.***iv\_top***);  
**tv\_search\_home** = (TextView) view.findViewById(R.id.***tv\_search\_home***);  
**tv\_message\_home** = (TextView) view.findViewById(R.id.***tv\_message\_home***);  
**iv\_top**.setOnClickListener(**this**);  
**tv\_search\_home**.setOnClickListener(**this**);  
**tv\_message\_home**.setOnClickListener(**this**);

@Override  
**public void** onClick(View view) **{  
  
 switch** (view.getId()) **{  
  
 case** R.id.***iv\_top***: Toast.makeText(**context**,"回顶部",Toast.***LENGTH\_SHORT***).show();  
 **recy\_home**.scrollToPosition(0);  
 **break**;  
  
 **case** R.id.***tv\_search\_home***: Toast.makeText(**context**,"搜索",Toast.***LENGTH\_SHORT***).show();  
 **break**;  
  
 **case** R.id.***tv\_message\_home***: Toast.makeText(**context**,"消息",Toast.***LENGTH\_SHORT***).show();  
 **break**;  
 **}  
}**

// 不忘实现监听接口

**public class** HomeFragment **extends** BaseFragment **implements** View.OnClickListener {

}

### 使用OKHttpUtils联网请求数据

#### 项目地址

https://github.com/hongyangAndroid/okhttputils

#### 添加依赖

compile 'com.zhy:okhttputils:2.6.2'

#### 配置OkhttpClient

##### 在Application中调用initClient（）方法进行设置，参照下图



##### 在工程中新建Application类来完成上述要求的初始化

*// 作用：整个软件***public class** MyApplication **extends** Application **{** @Override  
 **public void** onCreate() **{  
 super**.onCreate();  
  
 initOKHttpClient();  
 **}  
  
 private void** initOKHttpClient() **{** OkHttpClient okHttpClient = **new** OkHttpClient.Builder()  
 *//.addInterceptor(new LoggerInterceptor("TAG"))* .connectTimeout(10000L, TimeUnit.***MILLISECONDS***)  
 .readTimeout(10000L, TimeUnit.***MILLISECONDS***)  
 *//其他配置* .build();  
  
 OkHttpUtils.initClient(okHttpClient);  
 **}  
}**

##### 在配置文件<application>中的首个位置声明

android:name=".app.MyApplication"

#### 使用OKHttpUtils，在HomeFragment中进行联网请求测试

##### 请求参考



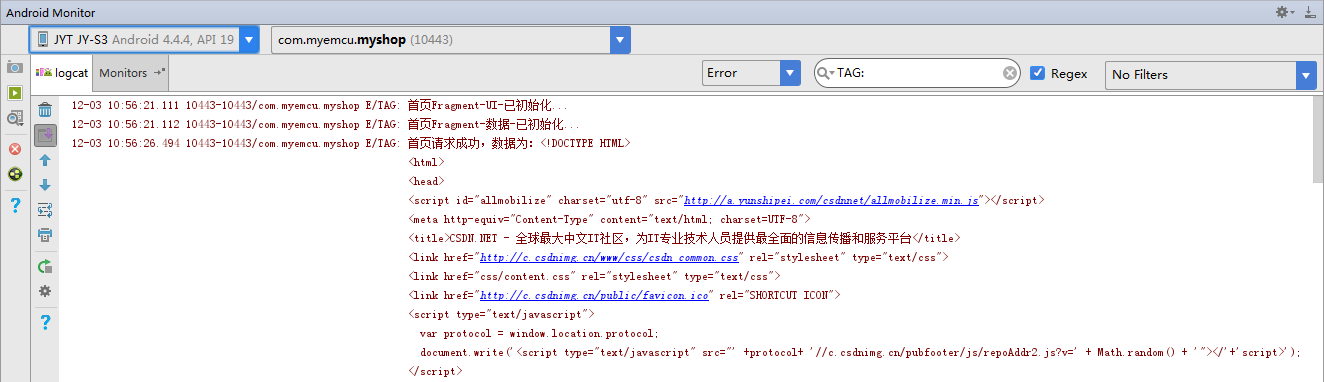
##### 具体实现

@Override  
**public void** initData() **{  
 super**.initData();  
 Log.e("TAG","首页Fragment-数据-已初始化...");  
  
 *// OKHttpUtils的Get请求* String url = "http://www.csdn.net/"; // 对该地址的数据进行联网请求  
 OkHttpUtils  
 .get()  
 .url(url)  
 */\*.addParams("username", "hyman")  
 .addParams("password", "123")\*/* .build()  
 .execute(**new** StringCallback()  
 **{** @Override *// 联网失败回调它* **public void** onError(Call call, Exception e, **int** id) **{** Log.e("TAG","首页请求失败，原因："+e.getMessage());  
 **}** @Override *// 联网成功回调它(response：请求成功的数据；id：区分 http(100) / https(101))* **public void** onResponse(String response, **int** id) **{** Log.e("TAG","首页请求成功，数据为："+response);  
 **}  
 }**);  
**}**

##### 添加权限

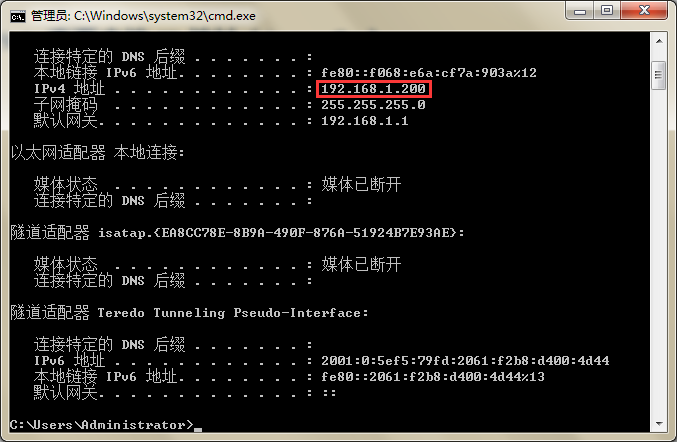
<**uses-permission** android:name="android.permission.INTERNET"/>

##### 测试结果（开WiFi）

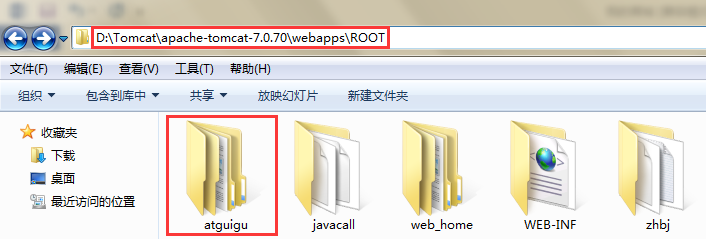


### 请求自己的数据

#### 查看本地IP地址（ipconfig）

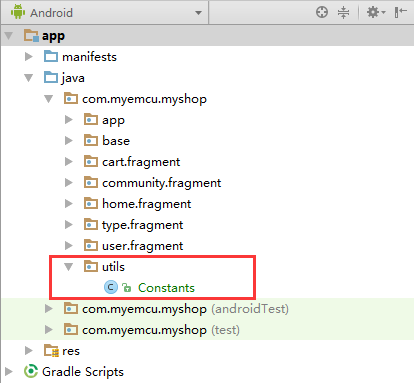


#### 将服务器资源文件复制到Tomcat的webapps\ROOT路径



#### 在工程中新建工具类-常量类

##### 工程目录树



##### 代码内容

*// 配置各页面联网地址***public class** Constants **{  
 public final static** String ***HOME\_URL*** = "http://192.168.1.200:8080/atguigu/json/HOME\_URL.json";  
**}**

#### 源码修改

String url = Constants.***HOME\_URL***;

#### 测试结果，略有延时（开WiFi、Tomcat）

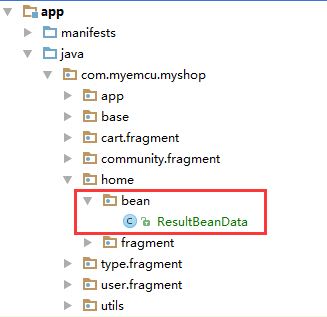


### FastJson解析数据

#### 添加依赖

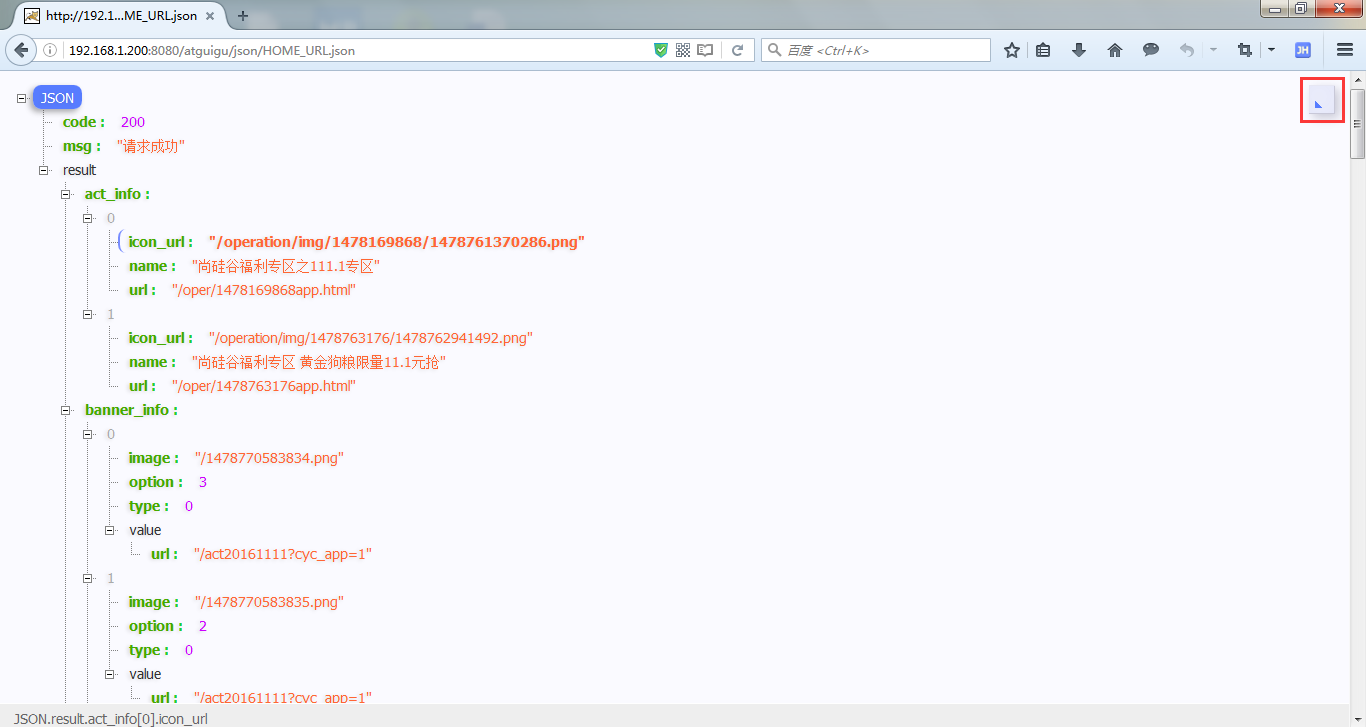
compile 'com.alibaba:fastjson:1.1.55.android'

#### 建立Bean对象



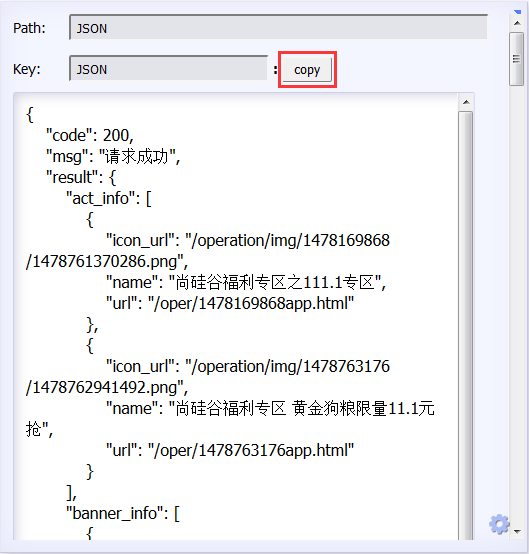
#### 在ResultBeanData中，用Gson（FastJson）生成bean代码（Alt+S）

##### IE(火狐)打开：<http://192.168.1.200:8080/atguigu/json/HOME_URL.json>

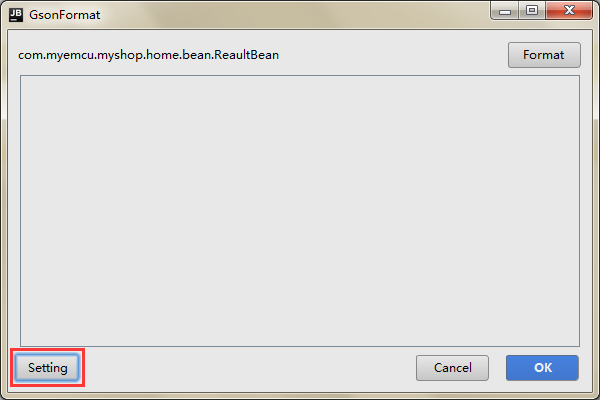


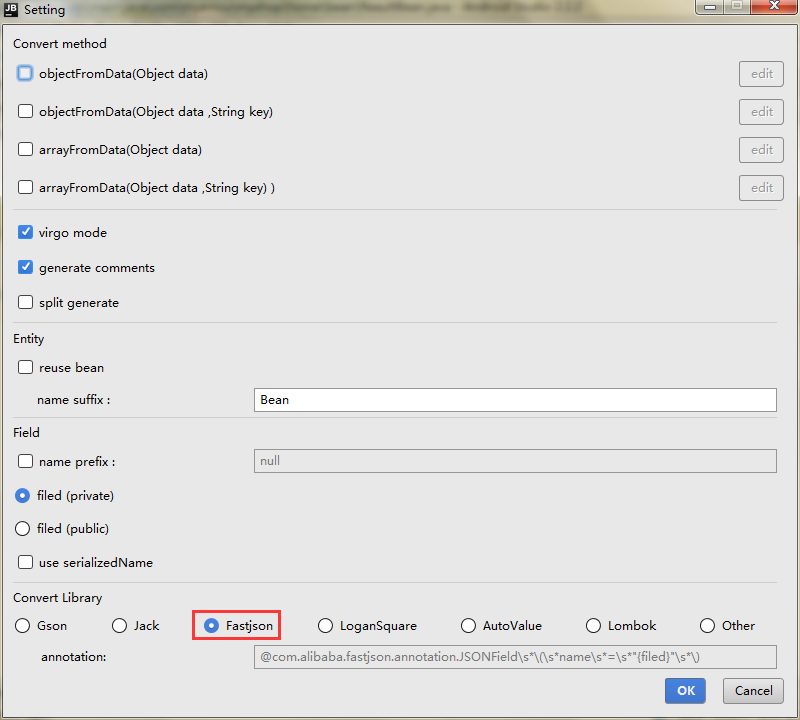
点这里

##### 展开Json(右上角)

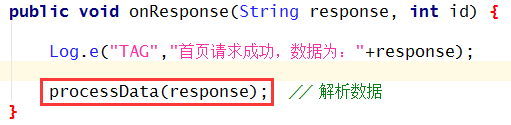


##### 将里面的内容复制到GsonFormat并使用FastJson生成Bean





#### 在HomeFragment中的联网成功处，解析数据



#### 解析代码（解析一条作测试）

**private void** processData(String json) **{** ResultBeanData resultBeanData = JSON.parseObject(json,ResultBeanData.**class**); *// 解析到后边的类中* **resultBean** = resultBeanData.getResult();  
 Log.e("TAG","解析成功=="+**resultBean**.getHot\_info().get(0).getName());  
**}**

#### 解析结果



### 首页UI

#### Json数据总览



#### UI出场顺序

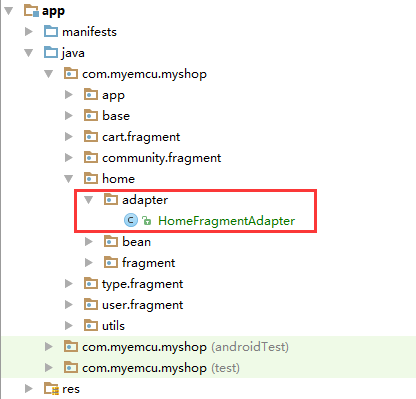
* 横幅（Banner） —— 0
* 频道（Channel） —— 1
* 活动（Act） —— 2
* 秒杀（Seckill） —— 3
* 推荐（Recommend） —— 4
* 热卖（Hot） —— 5

### 首页适配器(重点)

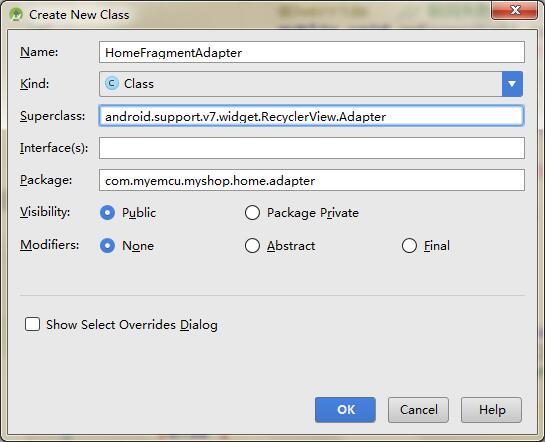
#### 总适配器(HomeFragmentAdapter)

一个基于HomeFragment的总适配器HomeFragmentAdapter，它继承自RecyclerView.Adapter。

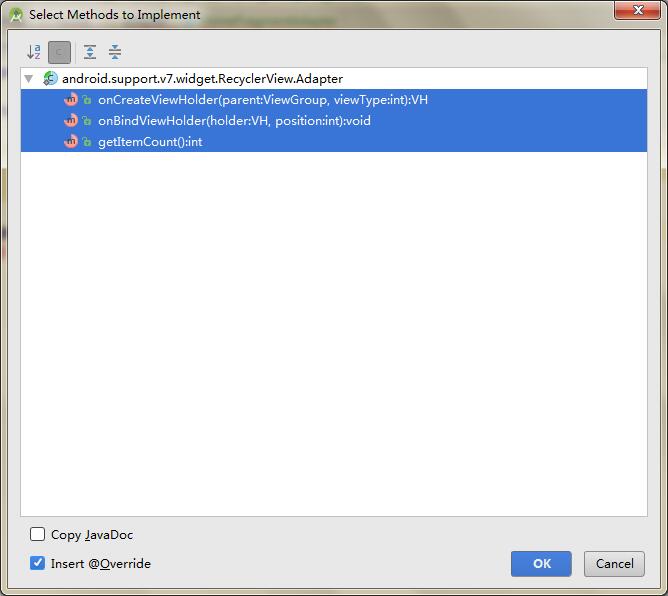
##### 建立适配器类



##### 继承对象



##### 实现方法



##### 原理分析

6种条目，对应6种RecyclerView，故需要一个继承于它的适配器：HomeFragmentAdapter，即：谁要显示RecyclerView，谁就需要适配器。

##### 实现步骤

1. 必须要实现的3个方法

@Override *// 创建ViewHolder(类似getView())***public** RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) **{  
 return null**;  
**}**@Override  
**public void** onBindViewHolder(RecyclerView.ViewHolder holder, **int** position) **{  
  
}**@Override *// 根据json可知，有6条item***public int** getItemCount() **{  
 return** 1;  
 *//return 6; // 一条条做，最后才是这个，为6就是返回0~5***}**

1. 变量定义

private final static int *BANNER* = 0; *// 横幅*private final static int *CHANNEL* = 1; *// 频道*private final static int *ACT* = 2; *// 活动*private final static int *SECKILL* = 3; *// 秒杀*private final static int *RECOMMEND* = 4; *// 推荐*private final static int *HOT* = 5; *// 热卖*private int currentType = *BANNER*; *// 定义当前条目的View类型(共6种)*

1. 补充实现的一个方法

@Override *// 获取条目的View类型(因为有6条)***public int** getItemViewType(**int** position) **{  
  
 switch** (position) **{  
 case *BANNER***: **currentType**=***BANNER***;  
 **break**;  
  
 **case *CHANNEL***: **currentType**=***CHANNEL***;  
 **break**;  
  
 **case *ACT***: **currentType**=***ACT***;  
 **break**;  
  
 **case *SECKILL***: **currentType**=***SECKILL***;  
 **break**;  
  
 **case *RECOMMEND***: **currentType**=***RECOMMEND***;  
 **break**;  
  
 **case *HOT***: **currentType**=***HOT***;  
 **break**;  
 **}  
  
 return currentType**;  
**}**

1. 到HomeFragment中的数据解析处设置该适配器

**private void** processData(String json) **{** ResultBeanData resultBeanData = JSON.parseObject(json,ResultBeanData.**class**); *// 解析到后边类中* **resultBean** = resultBeanData.getResult();  
 *//Log.e("TAG","解析成功=="+resultBean.getHot\_info().get(0).getName()); // 解析测试* **if** (**resultBean** != **null**) **{** *// 有数据，设置适配器* **adapter** = **new** HomeFragmentAdapter(**context**,**resultBean**); *// 传入上下文和数据* **recy\_home**.setAdapter(**adapter**);  
 **}else {  
  
 }  
}**

1. 适配器中的构造器

*// 构造器(上下文、数据、item)***public** HomeFragmentAdapter(Context context, ResultBeanData.ResultBean resultBean) **{  
 this**.**context**=context;  
 **this**.**resultBean**=resultBean;  
 **layoutInflater**=LayoutInflater.from(context); *// 用这个初始化布局***}**

1. 补充定义变量

*// 构造器接收***private final** Context **context**;  
**private final** ResultBeanData.ResultBean **resultBean**;  
*// 布局初始化***private final** LayoutInflater **layoutInflater**; *// 直接用View.inflate的性能消耗较大*

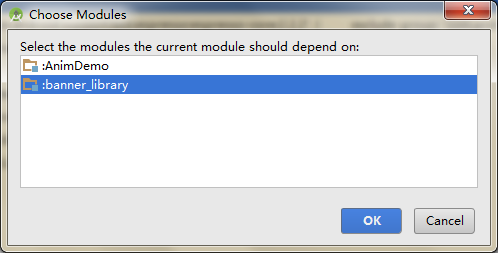
#### 横幅实现类(Banner)

##### 导入开源项目库banner

导入后名称改为banner\_library，路径：【…3 源码、课件\资料\1.第三方库\banner】



##### 添加Module依赖



##### 创建BannerViewHolder（即：创建ViewHolder）

@Override *// 创建ViewHolder(类似getView())，viewType来自getItemViewType(position)***public** RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) **{  
  
 if** (viewType == ***BANNER***) **{  
 return new** BannerViewHolder(**context**,**layoutInflater**.inflate(R.layout.***banner\_viewpager***,**null**));  
 **}  
  
 return null**;  
**}**

##### banner\_viewpager.xml

说明：命名空间app来自其库中的属性文件attr.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"   
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <**com.youth.banner.Banner** android:id="@+id/banner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="130dp"  
 app:indicator\_height="5dp"  
 app:indicator\_width="5dp"  
 app:is\_auto\_play="true" />  
</**LinearLayout**>

##### 实现BannerViewHolder内部类（暂无数据，数据在后边的setData()中）

**private class** BannerViewHolder **extends** RecyclerView.ViewHolder **{  
 private final** Context **context**;  
 **private final** Banner **banner**;  
  
 // 构造器(传入：上下文、布局文件)

// (接收：上下文、布局中的控件)

**public** BannerViewHolder(Context context, View itemview) **{** *// itemview：传入的布局名* **super**(itemview);  
 **this**.**context**=context;  
 **this**.**banner**= (Banner) itemview.findViewById(R.id.***banner***); *// 接收布局中的控件* **}  
}**

##### 绑定BannerViewHolder（即：绑定ViewHolder）

@Override  
**public void** onBindViewHolder(RecyclerView.ViewHolder holder, **int** position) **{  
 if** (getItemViewType(position)==***BANNER***) **{** BannerViewHolder bannerViewHolder = (BannerViewHolder) holder;  
 bannerViewHolder.setData(**resultBean**.getBanner\_info()); // 传入Banner数据  
 **}  
}**

##### 显示数据showData()

**public void** setData(List<ResultBeanData.ResultBean.BannerInfoBean> banner\_info) **{** *// 建立一个String集合，用来装图片地址* List<String> imagesUrl = **new** ArrayList<>();  
  
 **for** (**int** i =0; i<banner\_info.size(); i++) **{** String url = banner\_info.get(i).getImage(); *// 得到每一条图片地址* imagesUrl.add(url);  
 **}  
  
 banner**.setBannerAnimation(Transformer.*Accordion*); *// 手风琴动画特效* **banner**.setBannerStyle(BannerConfig.***CIRCLE\_INDICATOR***); *// 横幅加点指示* **banner**.setImages(imagesUrl, **new** OnLoadImageListener() **{** @Override  
 **public void** OnLoadImage(ImageView view, Object url) **{** *// Glide联网请求图片(使用jar包)  
 // 说明：此处将banner库中的glide版本由3.6.1->3.7.0，才能烧写通过，不然就报多重定义错* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE*** + url).into(view);  
 **}  
 }**);  
 **banner**.setOnBannerClickListener(**new** OnBannerClickListener() **{** *// Banner点击事件* @Override  
 **public void** OnBannerClick(**int** position) **{** Toast.makeText(**context**,"你点了第"+position+"条横幅",Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
**}**

##### 设置HomeFeagment中的RecyclerView布局方式

**recy\_home**.setLayoutManager(**new** GridLayoutManager(**context**,1)); *// 1列*

##### 实现效果



##### 编程思想

HomeFragmentAdapter适配器的编写流程

1. 在需要用到该适配器的文件中（HomeFragment）将其new出并设置

**adapter** = **new** HomeFragmentAdapter(**context**,**resultBean**); *// 传入上下文和数据***recy\_home**.setAdapter(**adapter**);  
**recy\_home**.setLayoutManager(**new** GridLayoutManager(**context**,1)); *// RecyclerView还需设置布局样式*

1. 确定该适配器应当继承哪个

**public class** HomeFragmentAdapter **extends** RecyclerView.Adapter **{**

**}**

1. 根据传入参数编写构造器

*// 构造器***public** HomeFragmentAdapter(Context context, ResultBeanData.ResultBean resultBean) **{  
 this**.**context**=context;  
 **this**.**resultBean**=resultBean;  
  
 **layoutInflater**=LayoutInflater.from(context); *// 根据传入的上下文，创建该适配器的布局加载对象***}**

1. 必须实现的3个方法和补充实现的一个方法
2. 创建ViewHolder——onCreateViewHolder(ViewGroup parent, int viewType)；
3. 绑定ViewHolder——onBindViewHolder(RecyclerView.ViewHolder holder, int position)；
4. item条目数量；
5. getItemViewType(int position)——根据位置，设置当前条目类型（从0开始），它的返回结果会回传到onCreateViewHolder的int viewType中；
6. 真正实现Banner显示的类：**BannerViewHolder()。**

各自代码如下：

**onCreateViewHolder()**

@Override *// 创建ViewHolder(类似getView())，viewType来自getItemViewType(position)***public** RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) **{  
  
 if** (viewType == ***BANNER***) **{** *// 传入上下文、布局(itemview)* **return new BannerViewHolder**(**context**, **layoutInflater**.inflate(R.layout.***banner\_viewpager***,**null**));  
 **}  
  
 return null**;  
**}**

**onBindViewHolder()**

@Override *// 绑定ViewHolder***public void** onBindViewHolder(RecyclerView.ViewHolder holder, **int** position) **{**

**if** (getItemViewType(position)==***BANNER***) **{** BannerViewHolder bannerViewHolder = (BannerViewHolder) holder;  
 bannerViewHolder.showData(**resultBean**.getBanner\_info()); *// 传入Banner数据* **}  
}**

**getItemCount()**

@Override *// 根据json可知，有6条item***public int** getItemCount() **{  
 return** 1;  
 *//return 6; // 一条条做，最后才是这个，为6就是返回0~5***}**

**getItemViewType()**

@Override *// 获取条目的View类型(因为有6条)***public int** getItemViewType(**int** position) **{  
  
 switch** (position) **{  
 case *BANNER***: **currentType**=***BANNER***;  
 **break**;  
  
 **case *CHANNEL***: **currentType**=***CHANNEL***;  
 **break**;  
  
 **case *ACT***: **currentType**=***ACT***;  
 **break**;  
  
 **case *SECKILL***: **currentType**=***SECKILL***;  
 **break**;  
  
 **case *RECOMMEND***: **currentType**=***RECOMMEND***;  
 **break**;  
  
 **case *HOT***: **currentType**=***HOT***;  
 **break**;  
 **}  
  
 return currentType**;  
**}**

**BannerViewHolder()**

**private class** BannerViewHolder **extends** RecyclerView.ViewHolder **{**

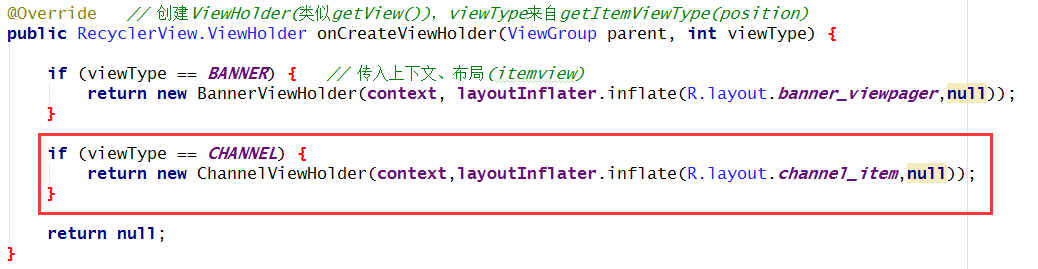
**private final** Context **context**;  
 **private final** Banner **banner**;  
  
 *// 构造器(传入：上下文、布局文件)  
 // (接收：上下文、布局中的控件)* **public** BannerViewHolder(Context context, View itemview) **{***// itemview：传入的布局名*

**super**(itemview);  
 **this**.**context**=context;  
 **this**.**banner**= (Banner) itemview.findViewById(R.id.***banner***);*// 接收布局中的控件* **}  
  
 public void** showData(List<ResultBeanData.ResultBean.BannerInfoBean> banner\_info) **{** *// 建立一个String集合，用来装图片地址* List<String> imagesUrl = **new** ArrayList<>();  
  
 **for** (**int** i =0; i<banner\_info.size(); i++) **{** String url = banner\_info.get(i).getImage(); *// 得到每一条图片地址* imagesUrl.add(url);  
 **}  
  
 banner**.setBannerAnimation(Transformer.*Accordion*); *// 手风琴动画特效* **banner**.setBannerStyle(BannerConfig.***CIRCLE\_INDICATOR***); *// 横幅加点指示* **banner**.setImages(imagesUrl, **new** OnLoadImageListener() **{** @Override  
 **public void** OnLoadImage(ImageView view, Object url) **{** *// Glide联网请求图片(使用jar包)  
 // 说明：此处将banner库中的glide版本由3.6.1->3.7.0，才能烧写通过，不然就报多重定义错* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE*** + url).into(view);  
 **}  
 }**);

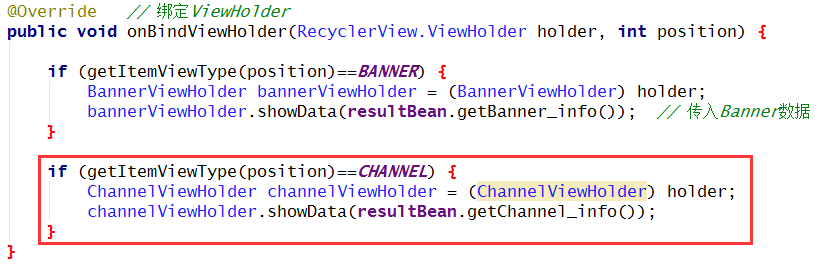
**banner**.setOnBannerClickListener(**new** OnBannerClickListener() **{** *// Banner点击事件* @Override  
 **public void** OnBannerClick(**int** position) **{** Toast.makeText(**context**,"你点了第"+position+"条横幅",Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
 **}  
}**

#### 频道适配器(Channel)

##### 在onCreateViewHolder()中追加



##### 在onBindViewHolder()中追加



##### ChannelViewHolder实现类

**private class** ChannelViewHolder **extends** RecyclerView.ViewHolder **{  
 private final** Context **context**;  
 **private** GridView **gv\_channel**;  
  
 **private** ChannelAdapter **adapter**;  
  
 **public** ChannelViewHolder(Context context, View itemView) **{  
 super**(itemView);  
  
 **this**.**context**=context;  
 **gv\_channel**= (GridView) itemView.findViewById(R.id.***gv\_channel***);  
 **}  
  
 public void** showData(**final** List<ResultBeanData.ResultBean.ChannelInfoBean> channel\_info) **{** *// 数据过来了，需要设置GridView适配器* **adapter** = **new** ChannelAdapter(**context**,channel\_info); *// 传入上下文和数据*

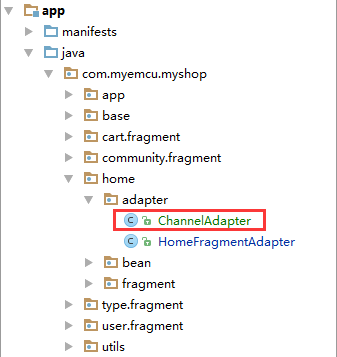
**gv\_channel**.setAdapter(**adapter**);  
 **gv\_channel**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() **{** @Override  
 **public void** onItemClick(AdapterView<?> view, View view1, **int** position, **long** l) **{** String content = channel\_info.get(position).getChannel\_name();  
 Toast.makeText(**context**,"你点击的是："+content,Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
 **}  
}**

##### ChannelAdapter类

**public class** ChannelAdapter **extends** BaseAdapter **{** *// GridView的适配器，用的是BaseAdapter* **private final** Context **context**;  
 **private final** List<ResultBeanData.ResultBean.ChannelInfoBean> **channel\_info**;  
  
 **public** ChannelAdapter(Context context, List<ResultBeanData.ResultBean.ChannelInfoBean>

channel\_info) **{  
 this**.**context**=context;  
 **this**.**channel\_info**=channel\_info;  
 **}** @Override  
 **public int** getCount() **{  
 return channel\_info**.size();  
 **}** @Override  
 **public** Object getItem(**int** i) **{  
 return null**;  
 **}** @Override  
 **public long** getItemId(**int** i) **{  
 return** 0;  
 **}  
  
 private static class** ViewHolder **{  
 private** ImageView **iv\_channel**;  
 **private** TextView **tv\_channel**;  
 **}** @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) **{** ViewHolder viewHolder;  
  
 **if** (convertView==**null**) **{** convertView = View.inflate(**context**,R.layout.***item\_channel***,**null**);  
 viewHolder = **new** ViewHolder();  
 viewHolder.**iv\_channel** = (ImageView) convertView.findViewById(R.id.***iv\_channel***);  
 viewHolder.**tv\_channel** = (TextView) convertView.findViewById(R.id.***tv\_channel***);  
 convertView.setTag(viewHolder);  
 **}else {** viewHolder = (ViewHolder) convertView.getTag();  
 **}** *// 显示图片* Glide.with(**context**)  
 .load(Constants.***BASE\_URL\_IMAGE***+**channel\_info**.get(position).getImage())  
 .into(viewHolder.**iv\_channel**);  
 *// 显示文字* viewHolder.**tv\_channel**.setText(**channel\_info**.get(position).getChannel\_name());  
  
 **return** convertView; *// 千万不要忘了这* **}  
}**

##### 工程目录



##### 实现效果



##### channel\_item.xml

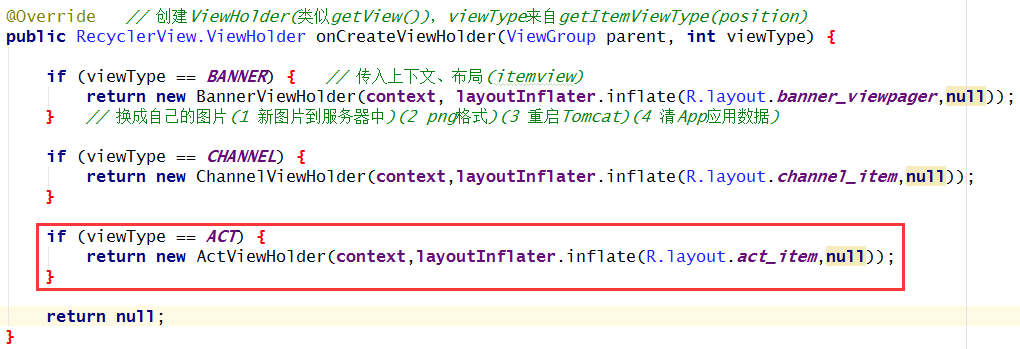
*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="170dp"  
 android:background="#fff"  
 android:orientation="vertical">  
  
 <**GridView** android:id="@+id/gv\_channel"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="150dp"  
 android:numColumns="5"></**GridView**>  
</**LinearLayout**>

##### item\_channel.xml

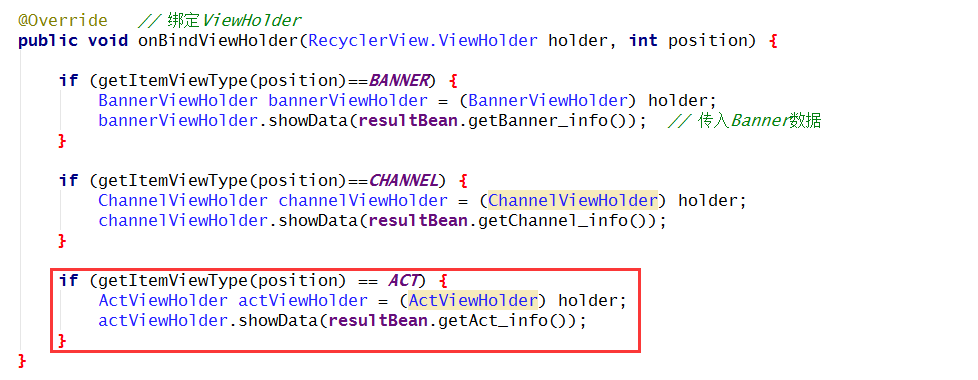
*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center\_horizontal"  
 android:orientation="vertical"  
 android:padding="10dp"  
 tools:ignore="UseCompoundDrawables">  
  
 <**ImageView** android:id="@+id/iv\_channel"  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:src="@drawable/menu\_cyc"  
 tools:ignore="ContentDescription"/>  
  
 <**TextView** android:id="@+id/tv\_channel"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:gravity="center\_horizontal"  
 android:text="@string/myGoods"  
 android:textColor="#000"  
 android:textSize="12sp"  
 tools:ignore="HardcodedText"/>  
</**LinearLayout**>

#### 活动适配器(Act)

##### ViewHolder视图



##### ViewHolder数据



##### ViewHolder实现类

*// 活动ViewHolder实现类***private class** ActViewHolder **extends** RecyclerView.ViewHolder **{  
 private final** Context **context**;  
 **private** ViewPager **act\_viewpager**;  
  
 **private** ActAdapter **adapter**;  
  
 **public** ActViewHolder(Context context, View itemView) **{** *// 要接收哪些就到其itemView中去看有些神马* **super**(itemView);  
  
 **this**.**context**=context;  
 **this**.**act\_viewpager**= (ViewPager) itemView.findViewById(R.id.***act\_viewpager***);  
 **}  
  
 public void** showData(**final** List<ResultBeanData.ResultBean.ActInfoBean> act\_info) **{** *// 数据已到位，设置适配器* **adapter** = **new** ActAdapter(**context**,act\_info);  
 **act\_viewpager**.setAdapter(**adapter**);  
 **act\_viewpager**.setPadding(10,10,10,10);  
 **}  
}**

##### 实现类中的适配器

*// ViewPager适配器***public class** ActAdapter **extends** PagerAdapter **{** *// PagerAdapter(Ctrl+Q，看至少需要实现几种方法)* **private final** Context **context**;  
 **private final** List<ResultBeanData.ResultBean.ActInfoBean> **act\_info**;  
  
  
 *// 构造器* **public** ActAdapter(Context context, List<ResultBeanData.ResultBean.ActInfoBean> act\_info) **{  
 this**.**context**=context;  
 **this**.**act\_info**=act\_info;  
 **}** @Override  
 **public int** getCount() **{  
 return act\_info**.size(); *// 返回总条数* **}** @Override  
 **public boolean** isViewFromObject(View view, Object object) **{  
 return** view == object; *// 套路(object，这是下边这方法的返回值)* **}** @Override *// container：ViewPager；position：对应页面位置* **public** Object instantiateItem(ViewGroup container, **final int** position) **{** *// 创建页面* ImageView img = **new** ImageView(**context**); *// 创建ImageView对象* img.setScaleType(ImageView.ScaleType.***FIT\_XY***); *// 设置拉伸* img.setOnClickListener(**new** View.OnClickListener() **{** @Override  
 **public void** onClick(View view) **{** String content = **act\_info**.get(position).getName();  
 Toast.makeText(**context**,content,Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
  
 *// 显示图片* Glide.with(**context**)  
 .load(Constants.***BASE\_URL\_IMAGE***+**act\_info**.get(position).getIcon\_url())  
 .into(img);  
  
 *// 添加到容器中(ViewPage自身)* container.addView(img);  
 **return** img;  
 **}** @Override  
 **public void** destroyItem(ViewGroup container, **int** position, Object object) **{** container.removeView((View) object);  
 **}  
}**

##### Act\_item.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<**FrameLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:clipChildren="true">  
  
 *<!--clipChildren中间大小-->* <**android.support.v4.view.ViewPager** android:id="@+id/act\_viewpager"  
 android:layout\_width="match\_parent"  
 android:layout\_height="120dp"  
 android:layout\_gravity="center"  
 android:clipChildren="true" />  
  
</**FrameLayout**>

##### ViewPager第三方美化

1. 添加依赖

compile 'com.zhy:magic-viewpager:1.0.1'

1. 具体使用

**public void** showData(**final** List<ResultBeanData.ResultBean.ActInfoBean> act\_info) **{  
 adapter** = **new** ActAdapter(**context**,act\_info); *// 数据已到位，设置适配器*  
 **act\_viewpager**.setAdapter(**adapter**);  
 **act\_viewpager**.setPadding(10,10,10,10);  
 *// ViewPager第三方美化* **act\_viewpager**.setOffscreenPageLimit(3); *// >=3* **act\_viewpager**.setPageTransformer(**true**, **new** RotateDownPageTransformer()); *// 扇形***}**

#### 秒杀适配器(Seckill)

##### onCreateViewHolder()

if (viewType == *SECKILL*) {  
 return new SeckillViewHolder(context,layoutInflater.inflate(R.layout.*seckill\_item*,null));  
}

##### onBindViewHolder()

**if** (getItemViewType(position) == ***SECKILL***) **{** SeckillViewHolder seckillViewHolder = (SeckillViewHolder) holder;  
 seckillViewHolder.showData(**resultBean**.getSeckill\_info());  
**}**

##### SeckillViewHolder的实现类

*// 秒杀viewHolder***private class** SeckillViewHolder **extends** RecyclerView.ViewHolder **{  
  
 private final** Context **context**;  
  
 **private** TextView **tv\_time\_seckill**,**tv\_more\_seckill**;  
 **private** RecyclerView **rv\_seckill**;  
  
 **private** SeckillAdapter **adapter**;  
  
 **public** SeckillViewHolder(Context context, View itemView) **{  
 super**(itemView);  
  
 **this**.**context**=context;  
  
 **tv\_time\_seckill** = (TextView) itemView.findViewById(R.id.***tv\_time\_seckill***);  
 **tv\_more\_seckill** = (TextView) itemView.findViewById(R.id.***tv\_more\_seckill***);  
 **rv\_seckill** = (RecyclerView) itemView.findViewById(R.id.***rv\_seckill***);  
  
 **}  
  
 public void** showData(**final** ResultBeanData.ResultBean.SeckillInfoBean seckill\_info) **{** *// 文本和RecyclerView的适配器* **adapter** = **new** SeckillAdapter(**context**,seckill\_info.getList()); *// 只需传入集合* **rv\_seckill**.setAdapter(**adapter**);  
 **rv\_seckill**.setLayoutManager(**new** *// 水平正序*

LinearLayoutManager(**context**,LinearLayoutManager.***HORIZONTAL***,**false**)); **}  
}**

##### Seckill适配器(RecyclerView单一ViewHolder)

**public class** SeckillAdapter **extends** RecyclerView.Adapter<SeckillAdapter.ViewHolder> **{** **private** ImageView **iv\_figure**;  
 **private** TextView **tv\_cover\_price**,**tv\_origin\_price**;**private final** Context **context**;  
 **private final** List<ResultBeanData.ResultBean.SeckillInfoBean.ListBean> **seckill\_infoList**;  
  
 **public class** ViewHolder **extends** RecyclerView.ViewHolder **{** *// 必须是public* **public** ViewHolder(View itemView) **{  
 super**(itemView);  
 **iv\_figure** = (ImageView) itemView.findViewById(R.id.***iv\_figure***);  
 **tv\_cover\_price** = (TextView) itemView.findViewById(R.id.***tv\_cover\_price***);  
 **tv\_origin\_price** = (TextView) itemView.findViewById(R.id.***tv\_origin\_price***);  
 **}  
 }** *// 构造器* **public** SeckillAdapter(Context context, List<ResultBeanData.ResultBean.SeckillInfoBean.ListBean> seckill\_infoList) **{  
 this**.**context**=context;  
 **this**.**seckill\_infoList**=seckill\_infoList;  
 **}** @Override  
 **public** ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) **{** View itemView = View.inflate(**context**, R.layout.***item\_seckill***,**null**);  
 **return new** ViewHolder(itemView);  
 **}** @Override  
 **public void** onBindViewHolder(ViewHolder holder, **final int** position) **{** *// 1 根据位置得到对应数据* **final** ResultBeanData.ResultBean.SeckillInfoBean.ListBean seckill\_list = **seckill\_infoList**.get(position);  
 *// 2 绑定数据* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE***+seckill\_list.getFigure()).into(**iv\_figure**);  
 **tv\_cover\_price**.setText(seckill\_list.getCover\_price());  
 **tv\_origin\_price**.setText(seckill\_list.getOrigin\_price());  
 **}** @Override  
 **public int** getItemCount() **{  
 return seckill\_infoList**.size(); *// 得到集合大小* **}  
}**

##### 设置Seckill适配器(RecyclerView)适配器

**public void** showData(**final** ResultBeanData.ResultBean.SeckillInfoBean seckill\_info) **{  
 adapter** = **new** SeckillAdapter(**context**,seckill\_info.getList()); *// 只需传入集合* **rv\_seckill**.setAdapter(**adapter**);  
 **rv\_seckill**.setLayoutManager(**new** LinearLayoutManager(**context**,LinearLayoutManager.***HORIZONTAL***,**false**)); *// 水平正序***}**

##### 实现效果



秒杀（1.12.5）

活动（1.12.4）

频道（1.12.3）

##### 自定义RecyclerView的item监听

* RecyclerView没有点击事件，要在其适配器中设置；
* seckill\_item.xml中，原例宽度为380dp导致横向图片的最后一张只能显示半张，应改为match\_parent。

1. 定义点击事件接口

**public interface** OnSeckillRecyclerView **{**

*// 某条被点击的时候回调，右边为传入参数* **void** onItemClick(**int** position, List<ResultBeanData.ResultBean.SeckillInfoBean.ListBean>

seckill\_infoList); **}**

1. 定义接口对象

private OnSeckillRecyclerView onSeckillRecyclerView;

1. setter接口对象(Alt+Insert)

**public void** setOnSeckillRecyclerView(OnSeckillRecyclerView onSeckillRecyclerView) **{  
 this**.**onSeckillRecyclerView** = onSeckillRecyclerView;  
**}**

1. ViewHolder(View itemView)中监听itemView

itemView.setOnClickListener(**new** View.OnClickListener() **{** @Override  
 **public void** onClick(View view) **{** *// 接口对象判空* **if** (**onSeckillRecyclerView** != **null**) **{  
 onSeckillRecyclerView**.onItemClick(getLayoutPosition(),**seckill\_infoList**); *// 与 1 呼应* **}  
 }  
}**);

1. 适配器外（showData内）调用

**adapter**.setOnSeckillRecyclerView(**new** SeckillAdapter.OnSeckillRecyclerView() **{** @Override  
 **public void** onItemClick(**int** position, List<ResultBeanData.ResultBean.SeckillInfoBean.ListBean>

seckill\_infoList) **{** String content = seckill\_info.getList().get(position).getName();  
 Toast.makeText(**context**,content,Toast.***LENGTH\_SHORT***).show();  
 **}  
}**);

##### Seckill倒计时 (该ViewHolder类中实现Handler接收)

思想：从服务端读取起始于结束时间，并让其相减得到秒杀时间。

1. 得到差值时间

**time\_dt**=Integer.valueOf(seckill\_info.getEnd\_time()) – Integer.valueOf(seckill\_info.getStart\_time());

1. 1s后发送消息

**handler**.sendEmptyMessageDelayed(0,1000); *// 无标志，1s*

1. Handler接收处理（其ViewHolder实现类中）

**private** Handler **handler** = **new** Handler() **{** *// 更新每秒秒杀时间*@Override  
 **public void** handleMessage(Message msg) **{  
 super**.handleMessage(msg);  
 **time\_dt**-=1000; *// 递减1s  
  
 // 系统转换有错误，小时位总有个8  
 /\*SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss"); // 总ms毫秒值格式转换  
 String time = format.format(new Date(time\_dt));\*/* **sec**= (**int**) (**time\_dt**/1000); *// 总ms转总s* **sec\_to\_hh\_mm\_ss();** ***// 总秒数转时分秒*****tv\_time\_seckill**.setText(**str\_hour**+":"+**str\_min**+":"+**str\_sec**);  
  
 *// 减后再发消息* **handler**.removeMessages(0);  
 **handler**.sendEmptyMessageDelayed(0,1000); *// 无标志，1s* **if** (**time\_dt** <= 0 ) **{** *// 秒杀结束* **handler**.removeCallbacksAndMessages(**null**);  
 Toast.makeText(**context**,"今日秒杀结束",Toast.***LENGTH\_SHORT***).show();  
 **}  
 }  
}**;

重要方法sec\_to\_hh\_mm\_ss();

private void sec\_to\_hh\_mm\_ss() {  
 hour=sec/3600;  
 min=(sec-hour\*3600)/60;  
 sec=(sec-hour\*3600)%60;  
  
 if (hour<10) {  
 str\_hour = "0"+hour+"";  
 }else {  
 str\_hour = hour+"";  
 }  
  
 if (min<10) {  
 str\_min = "0"+min+"";  
 }else {  
 str\_min = min+"";  
 }  
  
 if (sec<10) {  
 str\_sec = "0"+sec+"";  
 }else {  
 str\_sec = sec+"";  
 }  
}

##### SeckillViewHolder的完整实现

*// 秒杀viewHolder***private class** SeckillViewHolder **extends** RecyclerView.ViewHolder **{  
  
 private final** Context **context**;  
  
 **private** TextView **tv\_time\_seckill**,**tv\_more\_seckill**;  
 **private** RecyclerView **rv\_seckill**;  
  
 **private** SeckillAdapter **adapter**;  
  
 **private int hour**,**min**,**sec**;  
 **private** String **str\_hour**,**str\_min**,**str\_sec**;  
  
 *// 更新每秒秒杀时间* **private** Handler **handler** = **new** Handler() **{** @Override  
 **public void** handleMessage(Message msg) **{  
 super**.handleMessage(msg);  
  
 **time\_dt**-=1000; *// 递减1s  
  
 // 系统转换有错误，小时位总有个8  
 /\*SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss"); // 总ms毫秒值格式转换  
 String time = format.format(new Date(time\_dt));\*/* **sec**= (**int**) (**time\_dt**/1000); *// 总ms转总s* sec\_to\_hh\_mm\_ss(); *// 总秒数转时分秒* **tv\_time\_seckill**.setText(**str\_hour**+":"+**str\_min**+":"+**str\_sec**);  
  
 *// 减后再发消息* **handler**.removeMessages(0);  
 **handler**.sendEmptyMessageDelayed(0,1000); *// 无标志，1s* **if** (**time\_dt** <= 0 ) **{** *// 秒杀结束* **handler**.removeCallbacksAndMessages(**null**);  
 Toast.makeText(**context**,"今日秒杀结束",Toast.***LENGTH\_SHORT***).show();  
 **}  
 }  
 }**;  
  
 **public** SeckillViewHolder(Context context, View itemView) **{  
 super**(itemView);  
  
 **this**.**context**=context;  
  
 **tv\_time\_seckill** = (TextView) itemView.findViewById(R.id.***tv\_time\_seckill***);  
 **tv\_more\_seckill** = (TextView) itemView.findViewById(R.id.***tv\_more\_seckill***);  
 **rv\_seckill** = (RecyclerView) itemView.findViewById(R.id.***rv\_seckill***);  
 **}  
  
 public void** showData(**final** ResultBeanData.ResultBean.SeckillInfoBean seckill\_info) **{** *// 文本和RecyclerView的适配器* **adapter** = **new** SeckillAdapter(**context**,seckill\_info.getList()); *// 只需传入集合* **rv\_seckill**.setAdapter(**adapter**);  
 **rv\_seckill**.setLayoutManager(**new**

LinearLayoutManager(**context**,LinearLayoutManager.***HORIZONTAL***,**false**)); *// 水平正序  
  
 // item点击事件* **adapter**.setOnSeckillRecyclerView(**new** SeckillAdapter.OnSeckillRecyclerView() **{** @Override  
 **public void** onItemClick(**int** position,

List<ResultBeanData.ResultBean.SeckillInfoBean.ListBean> seckill\_infoList) **{** String content = seckill\_info.getList().get(position).getName();  
 Toast.makeText(**context**,content,Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
  
 *// 秒杀倒计时ms(范围由服务器上的起停实现来定)* **time\_dt**=Integer.valueOf(seckill\_info.getEnd\_time()) –

Integer.valueOf(seckill\_info.getStart\_time());  
 **handler**.sendEmptyMessageDelayed(0,1000); *// 无标志，1s* **}  
  
 private void** sec\_to\_hh\_mm\_ss() **{  
 hour**=**sec**/3600;  
 **min**=(**sec**-**hour**\*3600)/60;  
 **sec**=(**sec**-**hour**\*3600)%60;  
  
 **if** (**hour**<10) **{  
 str\_hour** = "0"+**hour**+"";  
 **}else {  
 str\_hour** = **hour**+"";  
 **}  
  
 if** (**min**<10) **{  
 str\_min** = "0"+**min**+"";  
 **}else {  
 str\_min** = **min**+"";  
 **}  
  
 if** (**sec**<10) **{  
 str\_sec** = "0"+**sec**+"";  
 **}else {  
 str\_sec** = **sec**+"";  
 **}  
 }  
}**

#### 推荐适配器(Recommend)

##### onCreateViewHolder()

if (viewType == *RECOMMEND*) {  
 return new RecommendViewHolder(context,layoutInflater.inflate(R.layout.*recommend\_item*,null));  
}

##### onBindViewHolder()

**if** (getItemViewType(position) == ***RECOMMEND***) **{** RecommendViewHolder recommendViewHolder = (RecommendViewHolder) holder;  
 recommendViewHolder.showData(**resultBean**.getRecommend\_info());  
**}**

##### RecommendViewHolder 实现类

*// RecommendViewHolder实现类(频道中有Gridview)* **private class** RecommendViewHolder **extends** RecyclerView.ViewHolder **{  
  
 private** TextView **tv\_more\_recommend**;  
 **private** GridView **gv\_recommend**;  
  
 **private final** Context **context**;  
  
 **public** RecommendViewHolder(Context context, View itemView) **{  
 super**(itemView);  
  
 **this**.**context**=context;  
  
 *// 实例化itemView* **tv\_more\_recommend** = (TextView) itemView.findViewById(R.id.***tv\_more\_recommend***);  
 **gv\_recommend** = (GridView) itemView.findViewById(R.id.***gv\_recommend***);  
 **}  
  
 public void** showData(**final** List<ResultBeanData.ResultBean.RecommendInfoBean> recommend\_info)

**{** *// 数据已到，设置GridView适配器* RecommendAdapter adapter = **new** RecommendAdapter(**context**,recommend\_info);**gv\_recommend**.setAdapter(adapter);

**gv\_recommend**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() **{** @Override  
 **public void** onItemClick(AdapterView<?> view, View view1, **int** position, **long** l) **{** String content = recommend\_info.get(position).getName();  
 Toast.makeText(**context**,content,Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
 **}  
 }  
}**

##### RecommendAdapter适配器

**public class** RecommendAdapter **extends** BaseAdapter **{** *// GridView所用的适配器* **private final** Context **context**;  
 **private final** List<ResultBeanData.ResultBean.RecommendInfoBean> **recommend\_info**;  
  
 **public** RecommendAdapter(Context context, List<ResultBeanData.ResultBean.RecommendInfoBean>

recommend\_info) **{  
 this**.**context**=context;  
 **this**.**recommend\_info**=recommend\_info;  
 **}** @Override  
 **public int** getCount() **{  
 return recommend\_info**.size();  
 **}** @Override  
 **public** Object getItem(**int** i) **{  
 return null**;  
 **}** @Override  
 **public long** getItemId(**int** i) **{  
 return** 0;  
 **}  
  
 private class** ViewHolder **{  
 private** ImageView **iv\_recommend**;  
 **private** TextView **tv\_name**,**tv\_price**;  
 **}** @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) **{** ViewHolder viewHolder = **new** ViewHolder();  
  
 **if** (convertView == **null**) **{** convertView = View.inflate(**context**, R.layout.***item\_recommend\_grid\_view***,**null**);  
 viewHolder.**iv\_recommend** = (ImageView) convertView.findViewById(R.id.***iv\_recommend***);  
 viewHolder.**tv\_name** = (TextView) convertView.findViewById(R.id.***tv\_name***);  
 viewHolder.**tv\_price** = (TextView) convertView.findViewById(R.id.***tv\_price***);  
 convertView.setTag(viewHolder);  
 **}else {** viewHolder= (ViewHolder) convertView.getTag();  
 **}** *// 接下来就该显示数据了(一个img，两个txt)* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE***+**recommend\_info**.get(position).getFigure())

.into(viewHolder.**iv\_recommend**);  
 viewHolder.**tv\_name**.setText(**recommend\_info**.get(position).getName());  
 viewHolder.**tv\_price**.setText("￥"+**recommend\_info**.get(position).getCover\_price());  
 **return** convertView;  
 **}  
}**

##### 实现效果



GridView(BaseAdapter)

#### 热卖适配器(Hot)

##### onCreateViewHolder()

if (viewType == *HOT*) {  
 return new HotViewHolder(context,layoutInflater.inflate(R.layout.*hot\_item*,null));  
}

##### onBindViewHolder()

**if** (getItemViewType(position) == ***HOT***) **{** HotViewHolder hotViewholder = (HotViewHolder) holder;  
 hotViewholder.showData(**resultBean**.getHot\_info());  
**}**

##### HotViewHolder 实现类

*// HotViewHolder实现类***private class** HotViewHolder **extends** RecyclerView.ViewHolder **{  
  
 private final** Context **context**;  
  
 **private** TextView **tv\_more\_hot**;  
 **private** GridView **gv\_hot**;  
  
 **public** HotViewHolder(Context context, View itemView) **{  
 super**(itemView);  
  
 **this**.**context**=context;  
  
 **tv\_more\_hot** = (TextView) itemView.findViewById(R.id.***tv\_more\_hot***);  
 **gv\_hot** = (GridView) itemView.findViewById(R.id.***gv\_hot***);  
 **}  
 public void** showData(**final** List<ResultBeanData.ResultBean.HotInfoBean> hot\_info) **{** HotAdapter adapter = **new** HotAdapter(**context**,hot\_info);  
 **gv\_hot**.setAdapter(adapter);  
 **gv\_hot**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() **{** @Override  
 **public void** onItemClick(AdapterView<?> view, View view1, **int** position, **long** l) **{** String content = hot\_info.get(position).getName();  
 Toast.makeText(**context**,content,Toast.***LENGTH\_SHORT***).show();  
 **}  
 }**);  
 **}  
}**

##### HotAdapter适配器

**public class** HotAdapter **extends** BaseAdapter **{  
  
 private final** Context **context**;  
 **private final** List<ResultBeanData.ResultBean.HotInfoBean> **hot\_info**;  
  
 **public** HotAdapter(Context context, List<ResultBeanData.ResultBean.HotInfoBean> hot\_info) **{  
 this**.**context**=context;  
 **this**.**hot\_info**=hot\_info;  
 **}** @Override  
 **public int** getCount() **{  
 return hot\_info**.size();  
 **}** @Override  
 **public** Object getItem(**int** i) **{  
 return null**;  
 **}** @Override  
 **public long** getItemId(**int** i) **{  
 return** 0;  
 **}  
  
 private class** ViewHolder **{  
 private** ImageView **iv\_hot**;  
 **private** TextView **tv\_name**,**tv\_price**;  
 **}** @Override  
 **public** View getView(**int** position, View convetView, ViewGroup group) **{** ViewHolder viewHolder = **new** ViewHolder();  
  
 **if** (convetView == **null**) **{** convetView = View.inflate(**context**, R.layout.***item\_hot\_grid\_view***,**null**);  
 viewHolder.**iv\_hot** = (ImageView) convetView.findViewById(R.id.***iv\_hot***);  
 viewHolder.**tv\_name** = (TextView) convetView.findViewById(R.id.***tv\_name***);  
 viewHolder.**tv\_price** = (TextView) convetView.findViewById(R.id.***tv\_price***);  
 convetView.setTag(viewHolder);  
 **}else {** viewHolder= (ViewHolder) convetView.getTag();  
 **}** *// 显示数据(一个img，两个txt)* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE*** +

**hot\_info**.get(position).getFigure()).into(viewHolder.**iv\_hot**);

viewHolder.**tv\_name**.setText(**hot\_info**.get(position).getName());  
 viewHolder.**tv\_price**.setText("￥"+**hot\_info**.get(position).getCover\_price());  
  
 **return** convetView;  
 **}  
}**

##### 踩坑说明

布局文件hot\_item.xml中的GridView高度为1100dp，原为1020dp，导致显示不完整。

##### 效果展示

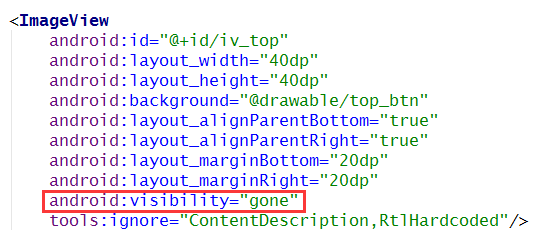


GridView(BaseAdapter)

### 首页上下滑动监听

功能:滑动到底部才出现向上（回顶部）按钮。

#### fragment\_home.xml中设置隐藏



#### HomeFragment中的代码

**private void** processData(String json) **{** ResultBeanData resultBeanData = JSON.parseObject(json,ResultBeanData.**class**); *// 解析到后边的类中* **resultBean** = resultBeanData.getResult();  
 *//Log.e("TAG","解析成功=="+resultBean.getHot\_info().get(0).getName());* **if** (**resultBean** != **null**) **{** *// 数据来了，设置RecyclerView适配器* **adapter** = **new** HomeFragmentAdapter(**context**,**resultBean**); *// 传入上下文和数据* **recy\_home**.setAdapter(**adapter**);  
 GridLayoutManager manager = **new** GridLayoutManager(**context**, 1);  
 **recy\_home**.setLayoutManager(manager); *// 1列  
 // 设置跨度大小的监听(GridView独有，Linearlayout无)* manager.setSpanSizeLookup(**new** GridLayoutManager.SpanSizeLookup() **{** @Override  
 **public int** getSpanSize(**int** position) **{** Log.e("TAGpos","position=="+position);  
 **if** (position>=5) **{  
 iv\_top**.setVisibility(View.***VISIBLE***);  
 **}else {  
 iv\_top**.setVisibility(View.***GONE***);  
 **}  
 return** 1;  
 **}  
 }**);  
 **}  
}**

### 跳转商品详情页

本节实现的是点击在首页中【今日秒杀】、【新品推荐】与【商城热卖】的内容实现商品页的跳转。

#### 页面布局



线性水平布局

（叠加显示）

线性垂直布局

帧布局

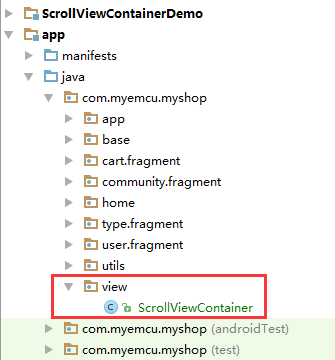
标题栏

##### 帧布局组成

1. 线性垂直布局（套一个ScrollViewContainer容器）
2. 线性水平布局（联系客服、收藏、购物车、加入购物车）
3. 包含更多布局（include layout="@layout/more\_layout"/）

##### ScrollViewContainer容器

此之为第三方开源项目，须先导入，然后再复制其类到本工程中即可引用。该项目的作用是将一个较长的页面划分成几个部分，每一部分滑完时停顿，然后再继续往下滑（本工程为两个部分）。



该部分代码如下

<**com.myemcu.myshop.view.ScrollViewContainer** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 *<!--ScrollView1-->* <**ScrollView** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:scrollbars="none">  
  
 <**LinearLayout** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#fff"  
 android:orientation="vertical"  
 tools:ignore="ScrollViewSize">  
  
 <**ImageView** android:id="@+id/iv\_good\_info\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="260dp"  
 android:layout\_gravity="center\_horizontal"  
 tools:ignore="ContentDescription"/>  
  
 <**TextView** android:id="@+id/tv\_good\_info\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="15dp"  
 android:text="【预售】《来年就交给你啦》限量礼盒版"  
 android:textColor="#323427"  
 tools:ignore="HardcodedText,RtlHardcoded"/>  
  
 <**TextView** android:id="@+id/tv\_good\_info\_desc"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="10dp"  
 android:text=" 预售截止 10 月 15 日，预售期为限量礼盒版；预售截止 10 月 15 日，预售期

为限量礼盒版；预售截止 10 月 15 日，预售期为限量礼盒版；重要的事情说3遍。"  
 android:textColor="#767f86"

tools:ignore="HardcodedText"/>  
  
 <**TextView** android:id="@+id/tv\_good\_info\_price"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="10dp"  
 android:text="￥150.00"  
 android:textColor="#ed3f3f"  
 android:textSize="20sp"

tools:ignore="HardcodedText"/>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:layout\_margin="10dp"  
 android:background="#99000000" />  
  
 <**LinearLayout** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="10dp">  
  
 <**TextView** android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="由"  
 android:textColor="#3c3d40" tools:ignore="HardcodedText"/>  
  
 <**TextView** android:id="@+id/tv\_good\_info\_store"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="老婆大人"  
 android:textColor="#ff4040" tools:ignore="HardcodedText"/>  
  
 <**TextView** android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="安排发货"  
 android:textColor="#3c3d40" tools:ignore="HardcodedText"/>  
  
 </**LinearLayout**>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="8dp"  
 android:background="#99000000" />  
  
 <**TextView** android:id="@+id/tv\_good\_info\_style"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:drawableRight="@drawable/home\_arrow\_right"  
 android:padding="10dp"  
 android:text="请选择款式"  
 android:textColor="#3c3d40"

tools:ignore="HardcodedText,RtlHardcoded"/>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="8dp"  
 android:background="#99000000" />  
  
 <**TextView** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="10dp"  
 android:text="提示：普通商品享有质量问题 7 天退货，签收后 3 个工作日内可换货。其他特殊商品

（如手办）的售后以页面说明为准（继续往下滑^.^）。"  
 android:textColor="#3c3d40"  
 tools:ignore="HardcodedText"/>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="8dp"  
 android:background="#99000000" />  
  
 <**TextView** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:padding="10dp"  
 android:text="图文详情"  
 android:textColor="#3c3d40"  
 android:textSize="15sp" tools:ignore="HardcodedText"/>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:background="#99000000" />  
  
 </**LinearLayout**>  
  
 </**ScrollView**>  
  
 *<!--ScrollView2-->* <**ScrollView** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <**WebView** android:id="@+id/wb\_good\_info\_more"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:ignore="ScrollViewSize"/>  
  
 </**ScrollView**>  
  
</**com.myemcu.myshop.view.ScrollViewContainer**>

#### 商品页实例化

因为要实例化的控件众多，故用一个网页工具【findview初始化离线使用】来解决此问题。

#### 商品页的跳转

可以从首页的“秒杀”、“推荐”、“热卖”中点任一商品后跳转到商品页，跳转时需携带Bean数据信息。

##### 建立Bean类



##### 实现序列化（要在Intent中传该对象）

public class GoodsBean implements Serializable {

}

##### 定义Bean成员（参考“热卖”的网页json）

private String cover\_price,figure,name,product\_id; *// 价格，图片路径，名称，产品Id*

##### 生成取值赋值及toString()方法

Alt+Insert

##### 到“秒杀”、“推荐”、“热卖”的点击事件中，通过Intent发出该对象，以“热卖”为例

**private static final** String ***GOODSBEAN*** = "goodsbean"; *// Intent字段*

GoodsBean goodsBean = **new** GoodsBean(); *// “热卖”商品信息类*goodsBean.setCover\_price(hot\_info.get(position).getCover\_price());  
goodsBean.setFigure(hot\_info.get(position).getFigure());  
goodsBean.setName(hot\_info.get(position).getName());  
goodsBean.setProduct\_id(hot\_info.get(position).getProduct\_id());  
startGoodsInfoActivity(goodsBean);

*// 启动商品详情信息列表页***private void** startGoodsInfoActivity(GoodsBean goodsBean) **{** Intent intent = **new** Intent(**context**, GoodsInfoActivity.**class**);  
 intent.putExtra(***GOODSBEAN***,goodsBean);  
 **context**.startActivity(intent);  
**}**

##### 在商品页GoodsInfoActivity的onCreate中，接收该对象

**private static final** String ***GOODSBEAN*** = "goodsbean"; *// Intent字段*

*// 接收HomeFragmentAdapter的Intent数据***goodsBean** = (GoodsBean) getIntent().getSerializableExtra(***GOODSBEAN***);  
**if** (**goodsBean** != **null**) **{** Toast.makeText(**this**, "goodsBean=="+**goodsBean**.toString(), Toast.***LENGTH\_SHORT***).show();  
**}**

至此，商品页需要显示的商品数据就准备完毕。

#### 商品页的显示

##### 读取Bean对象

**goodsBean** = (GoodsBean) getIntent().getSerializableExtra(***GOODSBEAN***);

##### 判空并显示

**if** (**goodsBean** != **null**) **{**  
 showGoodsData(**goodsBean**);  
**}**

##### showGoodsData()方法

**private void** showGoodsData(GoodsBean goodsBean) **{**

*// 1 显示图片(iv\_good\_info\_image)，into到的是对象，不是Id* Glide.with(**this**).load(Constants.***BASE\_URL\_IMAGE***+goodsBean.getFigure()).into(**ivGoodInfoImage**);

*// 2 显示名称* **tvGoodInfoName**.setText(" "+goodsBean.getName()+"。");

*// 3 设置价格* **tvGoodInfoPrice**.setText("￥"+goodsBean.getCover\_price());

*// 4 显示WebView* showwebView(goodsBean.getProduct\_id());  
**}**

##### 嵌入WebView()方法

**private void** showWebView(String product\_id) **{**

**if** (product\_id != **null**) **{**

**wbGoodInfoMore**.loadUrl("http://www.baidu.com");

*// 设置JS支持* WebSettings webSettings = **wbGoodInfoMore**.getSettings();  
 webSettings.setJavaScriptEnabled(**true**);  
 webSettings.setCacheMode(WebSettings.***LOAD\_CACHE\_ELSE\_NETWORK***);  
  
 **wbGoodInfoMore**.setWebViewClient(**new** WebViewClient() **{** */\*@Override // 兼容低版本  
 public boolean shouldOverrideUrlLoading(WebView view, String url) {  
 view.loadUrl(url);  
 return true;  
 }\*/* @Override  
 **public boolean** shouldOverrideUrlLoading(WebView view, WebResourceRequest request) **{**

**if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***LOLLIPOP***) **{** view.loadUrl(request.getUrl().toString());  
 **}**

**return true**;  
 **}  
 }**);  
 **}  
}**

### 商品页Activity中的商品加入购物车，并在购物车Fragment中显示(难点)

#### 逻辑主线

##### 从商品页Activity中，添加到CartStorge购物车存储器类

CartStorge.getInstance().addData(**goodsBean**);

代码分析：

1. getInstance()：
2. 读取SharedPreferences本地数据并转换成List列表
3. 将List数据中的每一项，逐条写入sparseArray内存
4. addData()：
5. 从内存sparseArray中获取并添加数据
6. 重新同步到内存sparseArray中
7. 同步到本地SharedPreferences

##### 在CartFragment中，显示添加到购物车(CartStorge类)中的商品数据

List<GoodsBean> goodsBeanList = CartStorge.getInstance().getAllData(); *//getAllData()：从本地读出；getInstance()本地转内存***for** (**int** i = 0; i < goodsBeanList.size(); i++) **{** Log.e("TAGList",goodsBeanList.get(i).toString()); *// 得到列表中的每一项***}**

代码分析：

1. goodsBeanList

——列表对象

1. getInstance()：

走过场，仅牵出getAllData()的引用而已。

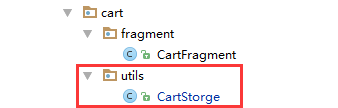
1. getAllData()：
2. 获取本地String(json)
3. String(json)转List

#### 实现步骤

##### Application中的全局Context

**private static** Context *context*;  
  
**public static** Context getContext() **{** *// Getter()方法* **return** *context*;  
**}**@Override  
**public void** onCreate() **{  
 super**.onCreate();  
  
 *context*=getApplicationContext(); // this也行  
  
 initOKHttpClient();  
**}**

##### 建立购物车utils类



##### 定义自身类对象(单例模式)

**private static** CartStorge *instance*;

##### 得到该类的实例

**public static** CartStorge getInstance() **{** *// 静态方法，外部直调* **if** (*instance* == **null**) **{** *instance* = **new** CartStorge(MyApplication.getContext()); *// 执行构造器，得到全局上下文* **}  
  
 return** *instance*;  
**}**

##### 构造器

**private** CartStorge(Context context) **{** *// 让构造器为private，这样该类就不会被实例化* **this**.**context**=context; *// 接收全局上下文* **sparseArray** = **new** SparseArray<>(); *// 视频中为sparseArray = new SparseArray<>(100);* listToSparseArray(); *// (本地转列表)，列表转内存***}**

##### listToSparseArray()

**private void** listToSparseArray() **{** // 列表转内存  
 // 1 读-----------------读取商品列表(所有商品信息)  
 List<GoodsBean> goodsBeanList = getAllData(); // (列表<--String)  
  
 // 2 存-----------------将列表保存到SparseArray内存 (内存<--列表)  
 **for** (**int** i = 0; i < goodsBeanList.size(); i++) **{** GoodsBean goodsBean = goodsBeanList.get(i); // 取列表中的每一项的位置  
 **sparseArray**.put(Integer.parseInt(goodsBean.getProduct\_id()),goodsBean);// 将产品Id放入内存  
 **}  
}**

##### getAllData()

**public** List<GoodsBean> getAllData() **{** *// 1 获取本地String* String json = CacheUtils.getString(**context**,***JSON\_CART***); *// 本地数据在Share持久化对象中存着的  
  
 // 2 String转化为List* List<GoodsBean> goodsBeanList = **new** ArrayList<>();  
  
 **if** (!TextUtils.isEmpty(json)) **{** *// 不为空才能解析(不然就报空指针异常崩溃)* goodsBeanList = **new** Gson().fromJson(json, **new** TypeToken<List<GoodsBean>>()**{}**.getType());  
 **}  
  
 return** goodsBeanList;  
**}**

##### 数据处理(增删改)

*// 增加数据***public void** addData(GoodsBean goodsBean) **{** *// 1 添加数据到内存中，即：sparseArray，若当前数据已存在，就修改成number递增* GoodsBean tempData = **sparseArray**.get(Integer.parseInt(goodsBean.getProduct\_id()));  
  
 **if** (tempData != **null**) **{** *// 内存中有了这条数据* tempData.setNumber(tempData.getNumber()+1);  
 **}else {** tempData=goodsBean;  
 tempData.setNumber(1); *// 至少要有一个* **}** *// 同步到内存中* **sparseArray**.put(Integer.parseInt(tempData.getProduct\_id()),tempData);  
  
 *// 2 同步到本地(说明：增删改查的时候都要同步到本地)* saveLocal();  
**}***// 删除数据***public void** delData(GoodsBean goodsBean) **{****sparseArray**.delete(Integer.parseInt(goodsBean.getProduct\_id())); *// 1 在内存中删除* saveLocal(); *// 2 删除后同步到本地*  
**}***// 更新数据***public void** updateData(GoodsBean goodsBean) **{** *// 1 内存中更新* **sparseArray**.put(Integer.parseInt(goodsBean.getProduct\_id()),goodsBean);  
 *// 2 同步到本地* saveLocal();  
**}**

##### 保存到本地

**private void** saveLocal() **{** *// 1 把sparseArray转换成List* List<GoodsBean> goodsBeanList = sparseToList();  
 *// 2 把List转换成string类型(使用Gson)* String json = **new** Gson().toJson(goodsBeanList);  
 *// 2 把String数据进行保存* CacheUtils.saveString(**context**,***JSON\_CART***,json);  
**}**

##### 内存转列表

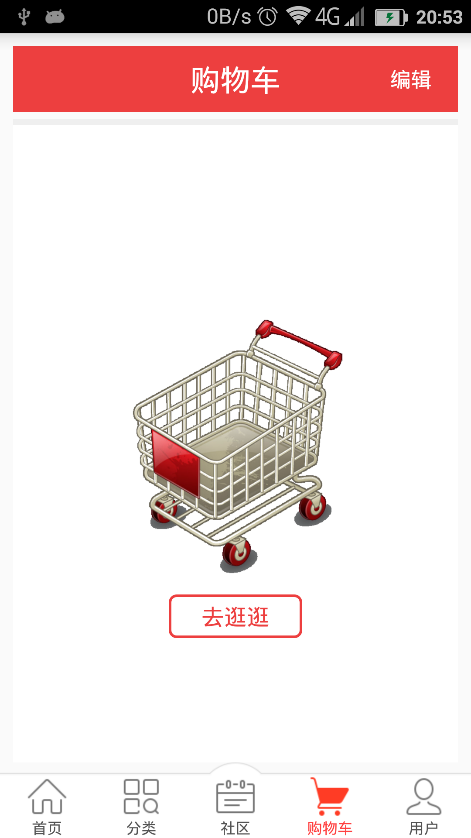
**private** List<GoodsBean> sparseToList() **{**

List<GoodsBean> goodsBeanList = **new** ArrayList<>();  
 **for** (**int** i = 0; i < **sparseArray**.size(); i++) **{** GoodsBean goodsBean = **sparseArray**.valueAt(i);  
 goodsBeanList.add(goodsBean);  
 **}  
 return** goodsBeanList;  
**}**

### 自定义控件实现增加、删除(直接参照AddSubView案例)

### 购物车CartFragment页面实现(最外边为线性垂直布局)

#### 商品结算、商品编辑、无商品效果图



#### 页头布局

<**RelativeLayout** android:padding="10dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginRight="10dp"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginBottom="5dp"  
 android:background="#ed3f3f"  
 tools:ignore="UselessParent">  
 <**TextView** android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginLeft="30dp"  
 android:layout\_weight="1"  
 android:text="购物车"  
 android:textColor="#ffffff"  
 android:textSize="20sp"  
 tools:ignore="HardcodedText,ObsoleteLayoutParam,RtlHardcoded"/>  
 <**TextView** android:id="@+id/tv\_shopcart\_edit"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_centerVertical="true"  
 android:layout\_marginRight="10dp"  
 android:text="编辑"  
 android:textColor="#ffffff"  
 tools:ignore="HardcodedText,RtlHardcoded"/>  
  
</**RelativeLayout**>

#### 分隔线

<**View** android:layout\_width="match\_parent"  
 android:layout\_height="5dp"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginRight="10dp"  
 android:background="#eeee" />

#### 帧布局

##### 有商品的线性布局(垂直)

1. RecyclerView(列表)

<**android.support.v7.widget.RecyclerView** android:id="@+id/rv\_cart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"/>

1. 页尾布局1(结算)

<**LinearLayout** android:id="@+id/ll\_check\_all"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#11000000"  
 android:orientation="horizontal"  
 android:visibility="visible">  
  
 <**CheckBox** android:id="@+id/checkbox\_all"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerVertical="true"  
 android:button="@null"  
 android:drawableLeft="@drawable/checkbox\_selector"  
 android:drawablePadding="10dp"  
 android:padding="10dp"  
 android:paddingLeft="0dp"  
 android:text="全选"  
 android:textColor="#303235"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText,ObsoleteLayoutParam,RtlHardcoded,RtlSymmetry"/>  
  
 <**TextView** android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:padding="10dp"  
 android:text="合计:"  
 android:textColor="#303235"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText"/>  
  
 <**TextView** android:id="@+id/tv\_shopcart\_total"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:padding="10dp"  
 android:text="￥0.00"  
 android:textColor="#ed3f3f"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText"/>  
  
 <**Button** android:id="@+id/btn\_check\_out"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="#ed3f3f"  
 android:text="去结算"  
 android:textColor="#fff"  
 tools:ignore="HardcodedText"/>  
</**LinearLayout**>

1. 页尾布局2(编辑)

<**LinearLayout** android:id="@+id/ll\_delete"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#11000000"  
 android:orientation="horizontal"  
 android:padding="2dp"  
 android:visibility="gone">  
  
 <**CheckBox** android:id="@+id/cb\_all"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerVertical="true"  
 android:layout\_weight="1"  
 android:button="@null"  
 android:drawableLeft="@drawable/checkbox\_selector"  
 android:drawablePadding="10dp"  
 android:padding="10dp"  
 android:paddingLeft="0dp"  
 android:text="全选"  
 android:textColor="#303235"  
 android:textSize="15sp"/>  
  
 <**Button** android:id="@+id/btn\_delete"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="35dp"  
 android:background="@drawable/words"  
 android:text="删除"  
 android:textColor="#303235"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText"/>  
  
 <**Button** android:id="@+id/btn\_collection"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="35dp"  
 android:layout\_marginLeft="15dp"  
 android:background="@drawable/wordsred"  
 android:text="收藏"  
 android:textColor="#ed3f3f"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText,RtlHardcoded"/>

</**LinearLayout**>

##### 无商品的空车布局(相对)

<**include** layout="@layout/empty\_cart"/>

<!--具体代码如下-->

*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/ll\_empty\_shopcart"  
 android:layout\_width="match\_parent"  
 android:background="#fff"  
 android:visibility="gone"  
 android:layout\_height="match\_parent">  
  
 <**RelativeLayout** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:ignore="UselessParent">  
  
 <**ImageView** android:id="@+id/iv\_empty"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_centerInParent="true"  
 android:background="@drawable/empty\_cart"  
 tools:ignore="ContentDescription"/>  
  
 <**TextView** android:id="@+id/tv\_empty\_cart\_tobuy"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/iv\_empty"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="15dp"  
 android:background="@drawable/shape\_rec\_textview"  
 android:paddingBottom="5dp"  
 android:paddingLeft="25dp"  
 android:paddingRight="25dp"  
 android:paddingTop="5dp"  
 android:text="去逛逛"  
 android:textColor="#ed3f3f"  
 android:textSize="15sp"  
 tools:ignore="HardcodedText"/>  
 </**RelativeLayout**>  
</**LinearLayout**>

#### CartFragment中initData()中的showData()方法

##### 读取数据

List<GoodsBean> **goodsBeanList** = CartStorge.getInstance().getAllData();

##### 显示数据

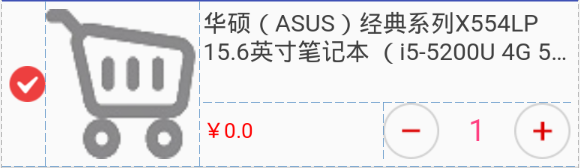
**if** (goodsBeanList != **null** && goodsBeanList.size()>0) **{** *// 有数据的情况(隐藏数据为空的布局)，设置Recyview适配器* **ll\_empty\_shopcart**.setVisibility(View.***GONE***);  
 **adapter** = **new** CartAdapter(**context**,goodsBeanList); *// 传入上下文和数据* **rv\_cart**.setAdapter(**adapter**);  
 *// RecyclerView布局管理器(上下文、方向、数据正序)* **rv\_cart**.setLayoutManager(**new** LinearLayoutManager(**context**,LinearLayoutManager.***VERTICAL***,**false**));  
**}else {** *// 无数据的情况(显示数据为空的布局)* **ll\_empty\_shopcart**.setVisibility(View.***VISIBLE***);  
**}**

##### 列表项适配器(RecyclerView的单一类型适配器)

**public class** CartAdapter **extends** RecyclerView.Adapter<CartAdapter.ViewHolder> **{  
  
 private final** Context **context**;  
 **private final** List<GoodsBean> **goodsBeanList**;  
  
 *// 适配器的构造器* **public** CartAdapter(Context context, List<GoodsBean> goodsBeanList) **{  
 this**.**context**=context;  
 **this**.**goodsBeanList**=goodsBeanList;  
 **}** *// 因为,Item只有一种类型，所以先写ViewHolder* **public class** ViewHolder **extends** RecyclerView.ViewHolder **{** *// 必须public* **private** CheckBox **cb\_gov**;  
 **private** ImageView **iv\_gov**;  
 **private** TextView **tv\_desc\_gov**,**tv\_price\_gov**;  
 **private** AddSubView **numberAddSubView**;  
  
 **public** ViewHolder(View itemView) **{  
 super**(itemView);  
  
 **cb\_gov** = (CheckBox) itemView.findViewById(R.id.***cb\_gov***);  
 **iv\_gov** = (ImageView) itemView.findViewById(R.id.***iv\_gov***);  
 **tv\_desc\_gov** = (TextView) itemView.findViewById(R.id.***tv\_desc\_gov***);  
 **tv\_price\_gov** = (TextView) itemView.findViewById(R.id.***tv\_price\_gov***);  
 **numberAddSubView** = (AddSubView) itemView.findViewById(R.id.***numberAddSubView***);  
 **}  
 }** @Override  
 **public** ViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) **{** View itemView = View.inflate(**context**, R.layout.***item\_shop\_cart***,**null**);  
 **return new** ViewHolder(itemView);  
 **}** @Override  
 **public void** onBindViewHolder(ViewHolder holder, **int** position) **{** *// 当item布局写完并在ViewHolder中实例化后就绑定数据  
 // 1 根据位置得到对应的Bean对象* GoodsBean goodsBean = **goodsBeanList**.get(position);  
 *// 2 设置数据* holder.**cb\_gov**.setChecked(goodsBean.isChecked()); *// 参数1(CheckBox)* Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE***+goodsBean.getFigure())

.into(holder.**iv\_gov**);  
 holder.**tv\_desc\_gov**.setText(goodsBean.getName());  
 holder.**tv\_price\_gov**.setText("￥"+goodsBean.getCover\_price());  
 **}** @Override  
 **public int** getItemCount() **{  
 return goodsBeanList**.size();  
 **}  
}**

#### Item项布局item\_shop\_cart.xml



*<?*xml version="1.0" encoding="utf-8"*?>*<**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <**LinearLayout** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="5dp">  
  
 <**CheckBox** android:id="@+id/cb\_gov"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:button="@drawable/checkbox\_selector"  
 android:checked="true"  
 android:clickable="false" />  
  
 <**ImageView** android:id="@+id/iv\_gov"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:background="@drawable/new\_img\_loading\_1"  
 android:scaleType="fitXY"  
 tools:ignore="ContentDescription"/>  
  
 <**LinearLayout** android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginLeft="5dp"  
 android:orientation="vertical"  
 tools:ignore="RtlHardcoded">  
  
 <**TextView** android:id="@+id/tv\_desc\_gov"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:ellipsize="end"  
 android:lines="2"  
 android:text="华硕（ASUS）经典系列X554LP 15.6英寸笔记本 （i5-5200U 4G 500G R5-M230 1G

独显 蓝牙 Win8.1 黑色）"  
 android:textColor="#404040"  
 android:textSize="16sp"  
 tools:ignore="HardcodedText"/>  
  
 <**LinearLayout** android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <**TextView** android:id="@+id/tv\_price\_gov"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="right"  
 android:layout\_weight="1"  
 android:gravity="center\_vertical"  
 android:text="￥0.0"  
 android:textColor="#f00"  
 android:textSize="14sp"  
 tools:ignore="HardcodedText"/>  
  
 <**com.myemcu.myshop.cart.view.AddSubView** android:id="@+id/numberAddSubView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content" />  
  
 </**LinearLayout**>  
  
 </**LinearLayout**>  
  
 </**LinearLayout**>  
  
 <**View** android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:background="#22000000" />  
  
</**LinearLayout**>

### 计算商品总价

计算从商品页添加的所有商品的总价，重复商品数量要更新显示到加减控件。

#### 向适配器中传入总价控件

adapter = new CartAdapter(context,goodsBeanList,tv\_price\_total);

#### 数据绑定中，设置商品数量

@Override  
**public void** onBindViewHolder(ViewHolder holder, **int** position) **{**

*// 当item布局写完并在ViewHolder中实例化后就绑定数据  
  
 // 1 根据位置得到对应的Bean对象* GoodsBean goodsBean = **goodsBeanList**.get(position);  
  
 *// 2 设置要显示的item数据* holder.**cb\_gov**.setChecked(goodsBean.isChecked());Glide.with(**context**).load(Constants.***BASE\_URL\_IMAGE***+goodsBean.getFigure()).into(holder.**iv\_gov**);holder.**tv\_desc\_gov**.setText(goodsBean.getName());holder.**tv\_price\_gov**.setText("￥"+goodsBean.getCover\_price()); *// 3 从商品页传来的商品数量* holder.**addSubView\_number**.setValue(goodsBean.getNumber()); *// 数量***}**

#### 在构造器中接收并显示总价

**public** CartAdapter(Context context, List<GoodsBean> goodsBeanList, TextView tv\_price\_total) **{  
  
 this**.**context**=context;  
 **this**.**goodsBeanList**=goodsBeanList;  
  
 **this**.**tv\_price\_total**=tv\_price\_total; // 接收总价  
 showTotalPrice(); *// 显示总价***}**

#### 显示商品总价

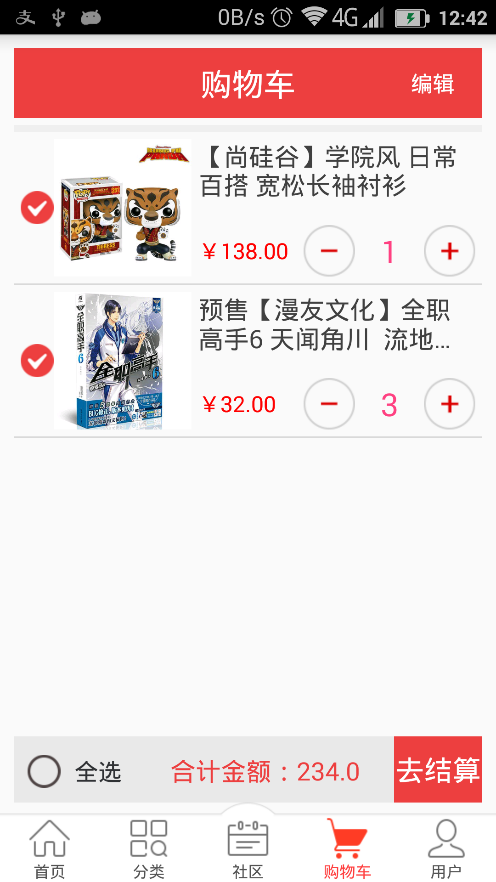
**private void** showTotalPrice() **{  
 tv\_price\_total**.setText("合计金额："+getTotalPrice());  
**}**

#### 获取商品总价

**private double** getTotalPrice() **{  
 double** totalPrice = 0; *// 初始化总价* **if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 for** (**int** i = 0; i < **goodsBeanList**.size() ; i++) **{** GoodsBean goodsBean = **goodsBeanList**.get(i); *// 得到列表中的每一项* **if** (goodsBean.isChecked()) **{** *// 计算选中项(做item时，默认就是选中状态)* totalPrice += Double.valueOf(goodsBean.getNumber()) \*

Double.valueOf(goodsBean.getCover\_price()); *// 数量\* 单价* **}  
 }  
 }  
 return** totalPrice;  
**}**

#### 实现效果(138\*1+32\*3=234)



#### 点按加减键，更新商品总价

##### 在适配器绑定方法中，进行AddSubView的回调监听

holder.**addSubView\_number**.setOnAddSubValueListener(**new** On…);

##### 更新列表中的商品数量

goodsBean.setNumber(value); *// 设置商品Bean中的商品数量*

##### 更新列表到内存与本地

CartStorge.getInstance().updateData(goodsBean); *// getInstance()单例方法*

##### 刷新适配器

notifyItemChanged(position);

##### 再次计算总价

showTotalPrice();

##### 完整代码

holder.**addSubView\_number**.setOnAddSubValueListener(**new** AddSubView.OnAddSubValueListener() **{** @Override  
 **public void** onAddSubValue(**int** value) **{** *// 1 回调数量更新到商品列表* goodsBean.setNumber(value); *// 2 更新列表到内存及本地* CartStorge.getInstance().updateData(goodsBean); *// getInstance()单例方法  
 // 3 刷新适配器* notifyItemChanged(position);  
 *// 4 再次计算总价* showTotalPrice();  
 **}  
}**);

#### 点按Item项的勾选，更新商品总价

##### 在适配器中设置Item项点击事件的接口回调

**public interface** OnItemClickListener **{  
 void** onItemClick(**int** position); *// 回传Item项的位置***}  
  
private** OnItemClickListener **onItemClickListener**; *// 接口对象***public void** setOnItemClickListener(OnItemClickListener onItemClickListener) **{** *// Setter方法* **this**.**onItemClickListener** = onItemClickListener;  
**}**

##### 在适配器ViewHolder中，设置itemView点击事件回传位置

itemView.setOnClickListener(**new** View.OnClickListener() **{**

@Override  
 **public void** onClick(View view) **{  
 if** (**onItemClickListener** != **null**) **{** *// 接口对象不为空* **onItemClickListener**.onItemClick(getLayoutPosition()); *// Item项的位置* **}  
 }  
}**);

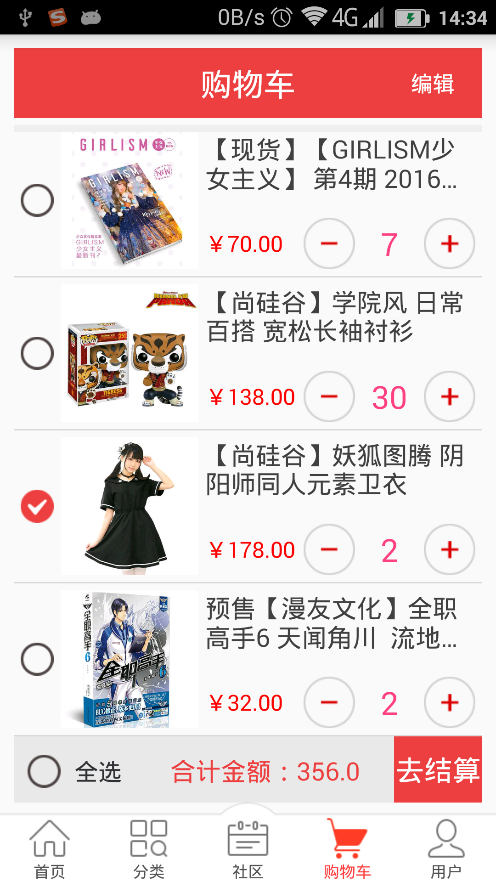
##### 在构造器中设置Item监听

**private void** setItemListener() **{**

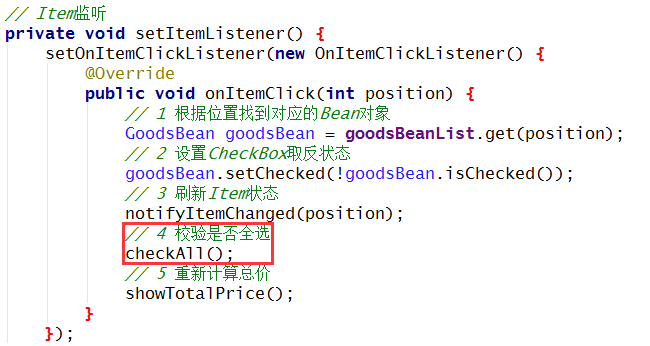
setOnItemClickListener(**new** OnItemClickListener() **{**

@Override  
 **public void** onItemClick(**int** position) **{** *// 1 根据位置找到对应的Bean对象* GoodsBean goodsBean = **goodsBeanList**.get(position);  
 *// 2 设置CheckBox取反状态* goodsBean.setChecked(!goodsBean.isChecked());  
 *// 3 刷新Item状态* notifyItemChanged(position);  
 *// 4 重新计算总价* showTotalPrice();  
 **}  
 }**);  
**}**

##### 实现效果



#### Item项目的全选与非全选更新下方全选状态



##### 实现代码

*// Item项的全选校验***private void** checkAll() **{  
 if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 int** cnt = 0;  
 **for** (**int** i = 0; i < **goodsBeanList**.size(); i++) **{** GoodsBean goodsBean = **goodsBeanList**.get(i); *// 从列表中得到某一具体的商品Bean* **if** (!goodsBean.isChecked()) **{** *// 只要有一个没选中  
 // 设置非全选* **check\_box\_all**.setChecked(**false**);  
 **}else {** cnt++; *// 选中的* **}  
 }  
 if** (cnt==**goodsBeanList**.size()) **{** *// 设置全选* **check\_box\_all**.setChecked(**true**);  
 **}  
 }  
}**

##### 实现效果



#### 下方CheckBox更新上方Item的全选与非全选状态

##### 构造器中设置点击事件

*// 设置全选Check\_Box的点击事件*check\_box\_all.setOnClickListener(**new** View.OnClickListener() **{** @Override  
 **public void** onClick(View view) **{** **boolean** isChecked = check\_box\_all.isChecked(); *// 1 得到全选Check\_Box状态* checkAll\_none(isChecked); *// 2 根据状态设置全选与非全选*  
 showTotalPrice(); *// 3 计算商品总价*  
 **}  
}**);

##### checkAll\_none(boolean isChecked)

*// 设置Item的全选和非全选***private void** checkAll\_none(**boolean** isChecked) **{  
 if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 for** (**int** i = 0; i < **goodsBeanList**.size(); i++) **{** GoodsBean goodsBean = **goodsBeanList**.get(i); *// 从列表中得到某一具体的商品Bean* goodsBean.setChecked(isChecked); *// isChecked为true，则每一项为true，反之亦然* notifyItemChanged(i);  
 **}  
 }  
}**

##### 实现效果



### CartFragment页面的编辑和完成状态的切换

#### 定义状态变量

private static final int *STATUS\_EDIT* = 1; *// 编辑状态*private static final int *STATUS\_FINISH* = 2; *// 完成状态*

#### 设置监听事件

tv\_cart\_edit.setTag(*STATUS\_EDIT*);  
tv\_cart\_edit.setOnClickListener(this);

#### 响应监听事件

**case** R.id.***tv\_cart\_edit***:  
 **int** action = (**int**) **tv\_cart\_edit**.getTag(); *// 获取状态*

**if** (action == ***STATUS\_EDIT***) **{** *// 切换到完成状态* showDelete();  
 **}else {** *// 切换为编辑状态* hideDelete();  
 **}**

**break**;

#### 完成状态

**private void** showDelete() **{**

*// 1 设置状态为完成* **tv\_cart\_edit**.setTag(***STATUS\_FINISH***);  
 **tv\_cart\_edit**.setText("完成");  
  
 *// 2 变成非勾选* **if** (**adapter** != **null**) **{  
 adapter**.checkAll\_none(**false**); *// 设置所有为非勾选* **}**

*// 3 删除视图的显示* **llDelete**.setVisibility(View.***VISIBLE***);

*// 4 结算视图的隐藏* **llCheckAll**.setVisibility(View.***GONE***);  
**}**

#### 编辑状态

**private void** hideDelete() **{**

*// 1 设置状态为编辑* **tv\_cart\_edit**.setTag(***STATUS\_EDIT***);  
 **tv\_cart\_edit**.setText("编辑");  
  
 *// 2 变成勾选* **if** (**adapter** != **null**) **{  
 adapter**.checkAll\_none(**true**); *// 设置所有为勾选* **}  
 check\_box\_all**.setChecked(**true**); *// 这也是* **adapter**.showTotalPrice(); *// 同上  
  
 // 3 删除视图的隐藏* **llDelete**.setVisibility(View.***GONE***);

*// 4 结算视图的显示* **llCheckAll**.setVisibility(View.***VISIBLE***);  
**}**

#### 实现效果

从左到右，依次为编辑态、完成态、重返编辑态。



### 完成态下的全选与非全选

#### 向适配器中增加参数

adapter = new CartAdapter(context,goodsBeanList,tv\_price\_total,check\_box\_all,cbAll);

#### 在适配器的checkAll()方法中设置全选和非全选

*// Item项的全选校验***private void** checkAll() **{  
 if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 int** cnt = 0;  
 **for** (**int** i = 0; i < **goodsBeanList**.size(); i++) **{** GoodsBean goodsBean = **goodsBeanList**.get(i); *// 从列表中得到某一具体的商品Bean* **if** (!goodsBean.isChecked()) **{** *// 只要有一个没选中  
 // 设置非全选* **check\_box\_all**.setChecked(**false**);  
 **cbAll**.setChecked(**false**);  
 **}else {** cnt++; *// 选中的* **}  
 }  
 if** (cnt==**goodsBeanList**.size()) **{** *// 设置全选* **check\_box\_all**.setChecked(**true**);  
 **cbAll**.setChecked(**true**);  
 **}  
 }else {  
 check\_box\_all**.setChecked(**false**);  
 **cbAll**.setChecked(**false**);  
 **}  
}**

#### 设置监听事件

*// 设置全选Check\_Box的点击事件(完成态)*cbAll.setOnClickListener(**new** View.OnClickListener() **{** @Override  
 **public void** onClick(View view) **{****boolean** isChecked = cbAll.isChecked(); *// 1 得到全选Check\_Box状态* checkAll\_none(isChecked); *// 2 根据状态设置全选与非全选*  
 **}  
}**);

#### 适配器外，编辑态与完成态的两个方法

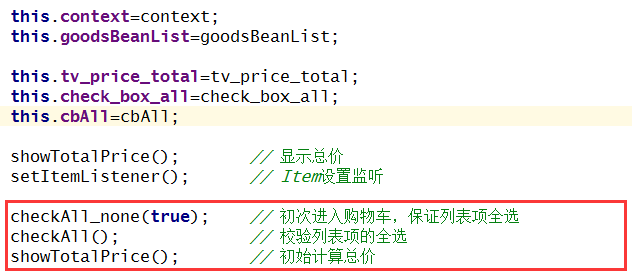
##### 完成态showDelete()方法

**private void** showDelete() **{** *// 1 设置状态为完成* **tv\_cart\_edit**.setTag(***STATUS\_FINISH***);  
 **tv\_cart\_edit**.setText("完成");  
 *// 2 变成非勾选* **if** (**adapter** != **null**) **{  
 adapter**.checkAll\_none(**false**); *// 设置所有为非勾选* **adapter**.checkAll();  
 **}** *// 3 删除视图的显示* **llDelete**.setVisibility(View.***VISIBLE***);  
 *// 4 结算视图的隐藏* **llCheckAll**.setVisibility(View.***GONE***);  
**}**

##### 编辑态hideDelete()方法

**private void** hideDelete() **{** *// 1 设置状态为编辑* **tv\_cart\_edit**.setTag(***STATUS\_EDIT***);  
 **tv\_cart\_edit**.setText("编辑");  
 *// 2 变成勾选* **if** (**adapter** != **null**) **{  
 adapter**.checkAll\_none(**true**); *// 设置所有为勾选* **adapter**.checkAll();  
 **}  
 check\_box\_all**.setChecked(**true**); *// 这也是* **adapter**.showTotalPrice(); *// 同上  
 // 3 删除视图的隐藏* **llDelete**.setVisibility(View.***GONE***);  
 *// 4 结算视图的显示* **llCheckAll**.setVisibility(View.***VISIBLE***);  
**}**

#### 初次进入购物车时的列表项与CheckBox的全选校验(注意顺序)



### 购物车页面最终状态转换图

#### 编辑态(初态)，初次进入购物车



#### 点上图中的“编辑”进入完成态



#### 点上图中的“完成”重返编辑态



### 完成态下RecyclerView-Item列表项的删除

#### CartFragment中监听删除键

**case** R.id.***btn\_delete***:  
 *// 1 删除选中项* **adapter**.deleteData();  
 *// 2 校验全选非全选* **adapter**.checkAll();  
 **break**;

adapter中的deleteData()方法实现：

*// 删除item项***public void** deleteData() **{  
 if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 for** (**int** i = 0; i < **goodsBeanList**.size() ; i++) **{** *// 删除选中项* GoodsBean goodsBean = **goodsBeanList**.get(i);  
 **if** (goodsBean.isChecked()) **{ // 在选中的情况下处理** *// 从List中移除* **goodsBeanList**.remove(goodsBean); *// remove是List<>中的方法  
 // 删除并更新到本地* CartStorge.getInstance().delData(goodsBean);  
 *// 刷新删除后的Item项* notifyItemRemoved(i);  
 *// 更新数量* **i--;**  
 **}  
 }  
 }  
}**

#### 编辑态Bug处理(购物车非空才能进入完成态)

**case** R.id.***tv\_cart\_edit***:

**int** action = (**int**) **tv\_cart\_edit**.getTag(); *// 获取状态*

**if** (action == ***STATUS\_EDIT***) **{** *// 切换到完成状态* **if** (**goodsBeanList**!=**null** && **goodsBeanList**.size()>0) **{** showDelete();  
 **}else {** Toast.makeText(**context**,"还是空的",Toast.***LENGTH\_SHORT***).show();  
 **}  
 }else {** *// 返回编辑态* hideDelete();  
 **}**

**break**;

#### 修改编辑态方法hideDelete()

**private void** hideDelete() **{**

*// 1 设置状态为编辑* **tv\_cart\_edit**.setTag(***STATUS\_EDIT***);  
 **tv\_cart\_edit**.setText("编辑");  
  
 *// 2 变成勾选* **if** (**goodsBeanList** != **null** && **goodsBeanList**.size()>0) **{  
 adapter**.checkAll\_none(**true**); *// 设置所有为勾选* **adapter**.checkAll();  
 **check\_box\_all**.setChecked(**true**); *// 这也是* **}  
  
 adapter**.showTotalPrice(); *// 同上  
  
 // 3 删除视图的隐藏* **llDelete**.setVisibility(View.***GONE***);  
 *// 4 结算视图的显示* **llCheckAll**.setVisibility(View.***VISIBLE***);  
**}**