# 修改超级微信项目为福利社项目

# 步骤1

1、将项目从github中导入到AS中

2、整理编译运行

3、切换为package目录结构，修改包名为fulicenter，两个复选框不勾选

4、切换为android目录结构，修改build.gradle(mudole)的applicationId

5、clean工程，rebuild project

6、将SuperWeChatApplication.java重命名为FuLiCenterApplication.java

7、注册环信服务器

echo "# fulicenter" >> README.md

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/clawpo/fulicenter.git

git push -u origin master

# 步骤2

根据服务器接口文档编写实体类

1. MessageBean已经存在，不用编写
2. UserBean已经存在，按照接口文档修改，去掉groups、latitude、longitude属性
3. ContactBean已经存在，按接口文档修改，去掉isGetMyLocation、isShowMyLocation属性
4. NewGoodBean,新品和精选的实体类,字段promote修改为isPromote,addTime类型为long，添加toString方法，为isPromote属性添加@JsonProperty("isPromote"),isPromote()方法添加@JsonIgnore标记，setPromote方法
5. BoutiqueBean，精品对象实体类，添加toString方法
6. CategoryGroupBean，大类别实体类，添加toString方法
7. CategoryChildBean，小类别实体类，添加toString方法
8. GoodDetailsBean,商品详情实体类，将albums和properties修改为AlbumBean[]和PropertyBean[]，添加toString方法，添加hashCode和equals方法，以goodId为比较基础，为isPromote属性添加@JsonProperty("isPromote"),isPromote()方法添加@JsonIgnore标记，setPromote方法
9. AlbumBean，相册实体类，添加toString方法
10. PropertyBean，属性实体类，添加toString方法
11. ColorBean,颜色实体类，添加toString方法
12. CollectBean,收藏产品实体类，添加toString方法，无参构造函数和有参构造函数(不包含id)
13. CartBean,购物车实体类，添加无参构造函数和有参构造函数(不包含GoodDetailBean)
14. 将以上所有实体类添加Serializable
15. 编译后将UserBean导致错误文件NetUtil、UserDao修改，删除无用的NearPeopleActivity和FindFragment中的引用

# 步骤3

根据服务器接口文档，整理NetUtil类

1. 删除所有跟群组相关的方法
2. 添加D文件，跟数据库表相关的各表字段
3. 修改I文件，删除群组相关，UserBean和UserDao中的定位字段，添加电商相关类和字段
4. 修改FuLiCenterApplication文件，删除群组，添加购物车
5. 集成第三方的库文件到src/main/jniLibs/目录下，并在gradle中添加依赖，如下：

compile files(**'src/main/jniLibs/alipaysdk.jar'**)  
compile files(**'src/main/jniLibs/alipaysecsdk.jar'**)  
compile files(**'src/main/jniLibs/alipayutdid.jar'**)  
compile files(**'src/main/jniLibs/libammsdk.jar'**)  
compile files(**'src/main/jniLibs/libone.jar'**)  
compile files(**'src/main/jniLibs/libpingpp.jar'**)  
compile files(**'src/main/jniLibs/pingpp-java-2.0.5.jar'**)  
compile files(**'src/main/jniLibs/UPPayAssistEx.jar'**)

1. 删除chatActivity中的mGroup和groupMembers属性，回退onGroupViewCreation的修改，删除DownloadGroupTask、GroupListener。
2. 删除MessageAdapter中MessageAdapter构造函数中的群聊代码
3. 将原始的GroupDetailsActivity、GroupAdapter、GroupActivity、NewGroupActivity、PublicGroupActivity回退到本工程中。
4. Utils中删除群聊和聊天室，DownloadContactsTask删除联系人群组下载，ChatAllHistoryFragment中获取群组的代码删除，SettingsFragment去掉群组
5. ChatAllHistoryAdapter中删除群组相关代码
6. MainActivity中的onApplicationAccept群组相关代码删除

# 步骤4

1. 修改MainAcivity，去掉没有登陆或者被T后跳转LoginActivity的代码
2. 新增变量UserBean mUser;并在onCreate中初始化
3. 去掉异步获取当前用户的昵称和头像的代码
4. 在setListener函数中，添加用户是否登陆的判断

**private void** setListener() {   
 **if** (**mUser** != **null**) {  
 *// setContactListener监听联系人的变化等* **contactListener** = **new** MyContactListener();  
 EMContactManager.*getInstance*().setContactListener(**contactListener**);  
  
 *// 注册一个监听连接状态的listener* **connectionListener** = **new** MyConnectionListener();  
 EMChatManager.*getInstance*().addConnectionListener(**connectionListener**);  
  
*// // 注册群聊相关的listener  
// groupChangeListener = new MyGroupChangeListener();  
// EMGroupManager.getInstance().addGroupChangeListener(groupChangeListener);* }  
 setMenuItemClickListener();  
 }

1. 在设置菜单项单击事件监听函数中，添加用户是否登陆的判断

if (!isConflict && !isCurrentAccountRemoved && mUser!=null) {

1. 在SplashActivity中修改用户是否登陆的逻辑判断

**protected void** onStart() {

**super**.onStart();  
 **new** Thread(**new** Runnable() {  
 **public void** run() {  
 **if** (DemoHXSDKHelper.*getInstance*().isLogined()) {  
 *// \*\* 免登陆情况 加载所有本地群和会话  
 //不是必须的，不加sdk也会自动异步去加载(不会重复加载)；  
 //加上的话保证进了主页面会话和群组都已经load完毕* **long** start = System.*currentTimeMillis*();  
 EMGroupManager.*getInstance*().loadAllGroups();  
 EMChatManager.*getInstance*().loadAllConversations();  
 **long** costTime = System.*currentTimeMillis*() - start;  
 *//等待sleeptime时长* **if** (***sleepTime*** - costTime > 0) {  
 **try** {  
 Thread.*sleep*(***sleepTime*** - costTime);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 *//将数据库中的当前登录用户保存在内存中* String userName = FuLiCenterApplication.*getInstance*().getUserName();  
 UserDao dao = **new** UserDao(SplashActivity.**this**);  
 UserBean user = dao.findUserByUserName(userName);  
 FuLiCenterApplication.*getInstance*().setUserBean(user);  
 *//添加连接服务器状态的判断  
// MessageBean msg = NetUtil.getServerStatus();  
// Log.e("SplashAvtivity","！！！！！！SplashAvtivity.NetUtil.login.user="+msg);  
// if(msg.isSuccess()){  
// Log.e("SplashAvtivity","！！！！！！SplashAvtivity.NetUtil.login.useif\*\*\*\*\*r="+msg.isSuccess());  
 //下载联系人数据* ArrayList<UserBean> contactList = FuLiCenterApplication.*getInstance*().getContactList();  
 **if**(contactList.size() == 0){  
 Log.*i*(**"main"**,**"SplashActivity.download contactList"**);  
 **new** DownloadContactsTask(SplashActivity.**this**, userName, 0, 20).execute();  
 }  
*// }else{  
// Log.e("SplashAvtivity","！！！！！！SplashAvtivity.NetUtil.login.userelse\*\*\*\*="+msg.isSuccess());  
// startActivity(new Intent(SplashActivity.this, LoginActivity.class));  
// NetUtil.isServerConnectioned();  
// finish();  
// }* }**else** {  
 **try** {  
 Thread.*sleep*(***sleepTime***);  
 } **catch** (InterruptedException e) {  
 }  
*// startActivity(new Intent(SplashActivity.this, LoginActivity.class));  
// finish();* }  
 *//进入主页面* startActivity(**new** Intent(SplashActivity.**this**, MainActivity.**class**));*//.putExtra("server\_status",msg.isSuccess()));* finish();  
 }  
 }).start();  
  
 }

1. 在value目录下的colors.xml中修改整体的颜色设置，统一修改为#fff

<color name="common\_bottom\_bar\_normal\_bg">#2d2f31</color>

<color name="top\_bar\_normal\_bg">#00acff</color>

<color name="bottom\_bar\_normal\_bg">#2D2F31</color>

新增颜色

<color name="bottom\_bar\_normal\_bg\_top">#C0BFBF</color>

<drawable name="menu\_item\_normal">#6A7077</drawable>

<drawable name="menu\_item\_pressed">#FF7A49</drawable>

<drawable name="menu\_item\_selected">#6A7077</drawable>

<color name="fragment\_bg">#eff2f6</color>  
 <color name="item\_good\_text">#313131</color>  
 <color name="item\_good\_price">#df3131</color>

1. 在style.xml中新增如下样式：

<style name="style\_menu\_btn">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">match\_parent</item>

<item name="android:padding">2dp</item>

<item name="android:textColor">@color/main\_botton\_text\_color</item>

<item name="android:textSize">10sp</item>

<item name="android:gravity">center</item>

</style>

<style name="style\_menu\_iv">

<item name="android:layout\_width">32dp</item>

<item name="android:layout\_height">32dp</item>

<item name="android:layout\_centerHorizontal">true</item>

<item name="android:layout\_marginTop">2dp</item>

</style>

<style name="style\_divide\_line">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">0.5dp</item>

<item name="android:background">@drawable/divide\_line\_bg</item>

</style>

<style name="style\_menu\_item">

<item name="android:layout\_width">0dp</item>

<item name="android:layout\_height">match\_parent</item>

<item name="android:layout\_weight">1</item>

<item name="android:layout\_marginTop">3dp</item>

<item name="android:layout\_marginBottom">3dp</item>

</style>

<style name="style\_menu\_layout">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">40dp</item>

<item name="android:layout\_marginBottom">5dp</item>

<item name="android:layout\_marginLeft">5dp</item>

<item name="android:layout\_marginRight">5dp</item>

<item name="android:layout\_marginTop">5dp</item>

<item name="android:background">#fff</item>

</style>

1. 修改activity\_main.xml布局底部菜单项为新品、精选、分类、购物车、我，其中新品代码如下：

<**RelativeLayout  
 android:id="@+id/layout\_new\_good"  
 style="@style/style\_menu\_item"**>  
 <**ImageView  
 android:id="@+id/ivNewGood"  
 style="@style/style\_menu\_iv"  
 android:src="@drawable/menu\_new\_good"**/>  
 <**TextView  
 android:id="@+id/btnNewGood"  
 style="@style/style\_menu\_btn"  
 android:layout\_below="@id/ivNewGood"  
 android:text="新品"**/>  
</**RelativeLayout**>

1. 编写menu\_new\_good,并保存到drawable目录下：

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<selector xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:state\_checked=*"true"*

android:drawable=*"@drawable/menu\_item\_new\_good\_selected"*/>

<item android:drawable=*"@drawable/menu\_item\_new\_good\_normal"*/>

</selector>

1. 新增menu\_item\_new\_good\_selected和menu\_item\_new\_good\_normal图片
2. 剩余的精选、分类、购物车、我类似，不再赘述

# 步骤5

1. 在MainActivity中添加以下变量

// 菜单项按钮

TextView mtvCartHint;

TextView mtvCart;

TextView mtvNewGood;

TextView mtvBoutique;

TextView mtvCategory;

TextView mtvPersonalCenter;

ImageView mivCart;

ImageView mivNewGood;

ImageView mivBoutique;

ImageView mivCategory;

ImageView mivPersonalCenter;

RelativeLayout mLayoutCart;

RelativeLayout mLayoutNewGood;

RelativeLayout mLayoutBoutique;

RelativeLayout mLayoutCategory;

RelativeLayout mLayoutPersonalCenter;

1. 在initView中初始化以上变量

mtvCartHint = getViewById(R.id.tvCartHint);

mtvCart = getViewById(R.id.tvCart);

mtvBoutique = getViewById(R.id.tvBoutique);

mtvCategory = getViewById(R.id.tvCategory);

mtvNewGood = getViewById(R.id.btnNewGood);

mtvPersonalCenter = getViewById(R.id.tvPersonalCenter);

mivBoutique=getViewById(R.id.ivBoutique);

mivCart=getViewById(R.id.ivCart);

mivCategory=getViewById(R.id.ivCategory);

mivNewGood=getViewById(R.id.ivNewGood);

mivPersonalCenter=getViewById(R.id.ivPersonalCenter);

mLayoutBoutique=getViewById(R.id.layout\_boutique);

mLayoutCart=getViewById(R.id.layout\_cart);

mLayoutCategory=getViewById(R.id.layout\_category);

mLayoutNewGood=getViewById(R.id.layout\_new\_good);

mLayoutPersonalCenter=getViewById(R.id.layout\_personal\_center);

# 步骤6

1. 编写商品适配器GoodAdapter继承BaseAdapter，复写父类的方法
2. 添加属性

Context context;  
 ArrayList<NewGoodBean> goodList;  
 ImageLoader imageLoader;  
 boolean misMore;  
 int sortBy;

1. 添加misMore的getters和setters方法，和构造函数。
2. 添加商品的排序方法sort

**private void** sort(**final int** sortBy){  
 Collections.*sort*(**goodList**, **new** Comparator<NewGoodBean>() {  
 @Override  
 **public int** compare(NewGoodBean g1, NewGoodBean g2) {  
 **int** result =0;  
 **switch** (sortBy){  
 **case** I.***SORT\_BY\_ADDTIME\_ASC***:  
 result = (**int**) (g1.getAddTime()-g2.getAddTime());  
 **break**;  
 **case** I.***SORT\_BY\_ADDTIME\_DESC***:  
 result = (**int**) (g2.getAddTime()-g1.getAddTime());  
 **break**;  
 **case** I.***SORT\_BY\_PRICE\_ASC***:  
 {  
 **int** p1 = convertPrice(g1.getCurrencyPrice());  
 **int** p2 = convertPrice(g2.getCurrencyPrice());  
 result = p1-p2;  
 }  
 **break**;  
 **case** I.***SORT\_BY\_PRICE\_DESC***:  
 {  
 **int** p1 = convertPrice(g1.getCurrencyPrice());  
 **int** p2 = convertPrice(g2.getCurrencyPrice());  
 result = p2-p1;  
 }  
 **break**;  
 }  
 **return** result;  
 }  
 **private int** convertPrice(String price){  
 price = price.substring(price.indexOf(**"￥"**)+1);  
 **int** p1 = Integer.*parseInt*(price);  
 **return** p1;  
 }  
 });  
}

1. 添加setSortBy、initItems、addItems方法来操作商品
2. 修改getCount、getItem方法
3. 创建item\_new\_good.xml来显示单个的商品
4. 添加ViewHolder来枚举布局文件中的控件

**class** ViewHolder{  
 LinearLayout **layoutGood**;  
 ImageView **ivGoodsThumb**;  
 TextView **tvGoodsName**;  
 TextView **tvCurrencyPrice**;  
 }

1. 修改getView方法

@Override  
**public** View getView(**int** position, View layout,**final** ViewGroup parent) {  
 ViewHolder holder = **null**;  
 **if**(layout==**null**){  
 layout= View.*inflate*(**context**, R.layout.***item\_new\_good***,**null**);  
 holder = **new** ViewHolder();  
 holder.**layoutGood** = (LinearLayout) layout.findViewById(R.id.***layoutGood***);  
 holder.**ivGoodsThumb**=(ImageView) layout.findViewById(R.id.***ivGoodThumb***);  
 holder.**tvGoodsName**=(TextView) layout.findViewById(R.id.***tvGoodName***);  
 holder.**tvCurrencyPrice**=(TextView) layout.findViewById(R.id.***tvCurrencyPrice***);  
 layout.setTag(holder);  
 }**else**{  
 holder = (ViewHolder) layout.getTag();  
 }  
 **final** NewGoodBean good = getItem(position);  
 holder.**tvCurrencyPrice**.setText(good.getCurrencyPrice());  
 holder.**tvGoodsName**.setText(good.getGoodsName());  
 String goodsThumb = good.getGoodsThumb();  
 holder.**ivGoodsThumb**.setTag(goodsThumb);  
 String savePath = goodsThumb.substring(goodsThumb.indexOf(**"/images"**)+1);  
 String uri = I.***SERVER\_ROOT*** +**"?"**+ I.***KEY\_REQUEST***+**"="**+ I.***REQUEST\_DOWNLOAD\_NEW\_GOOD*** +**"&"**+ I.***FILE\_NAME***+**"="**+goodsThumb;  
 Bitmap thumb = **imageLoader**.displayImage(uri, savePath, 150, 250, **new** ImageLoader.OnImageLoadListener() {  
 @Override  
 **public void** onSuccess(String path, Bitmap bitmap) {  
 ImageView iv = (ImageView) parent.getTag();  
 **if**(iv!=**null**){  
 iv.setImageBitmap(bitmap);  
 }  
 }  
  
 @Override  
 **public void** error(String errorMsg) {  
  
 }  
 });  
 **if**(thumb!=**null**){  
 holder.**ivGoodsThumb**.setImageBitmap(thumb);  
 }**else**{  
 holder.**ivGoodsThumb**.setImageResource(R.drawable.***nopic***);  
 }  
  
 holder.**layoutGood**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Log.*i*(**TAG**,**"new good item on click,good="**+good.toString());  
 **context**.startActivity(**new** Intent(**context**,GoodDetailsActivity.**class**)  
 .putExtra(D.NewGood.***KEY\_GOODS\_ID***,good.getGoodsId()));  
 }  
 });  
 **return** layout;  
}

# 步骤7

1. 创建NewGoodFragment类,继承Fragment，并重写onCreateView方法
2. 将PullRefreshView导入Utils包，将pull\_refresh\_view.xml导入layout，将pull\_push\_arrow\_down.png导入drawable-mdpi包中
3. 创建fragment\_new\_good.xml布局文件
4. 添加initView方法，初始化控件
5. 创建DownloadGoodsTask继承AsyncTask，并重写onPreExecute、doInBackground和onPostExecue方法
6. 添加属性，并重写有参构造函数

ProgressDialog **mdialog**;  
Context **mContext**;  
LoadStatus **loadStatus**;  
ArrayList<NewGoodBean> **mGoodList**;  
GoodAdapter **mAdapter**;  
ActionType **actionType**;  
**int catId**;  
*/\*\* 0:新品或者精选;1:分类 \*/***int goodType**;  
  
**public** DownloadGoodsTask(Context context, GoodAdapter mAdapter,  
 ArrayList<NewGoodBean> mGoodList, ActionType actionType, **int** catId, **int** goodType) {  
 **this**.**mContext** = context;  
 **this**.**mAdapter** = mAdapter;  
 **this**.**mGoodList** = mGoodList;  
 **loadStatus** = LoadStatus.***LOADING***;  
 **this**.**actionType** = actionType;  
 **this**.**catId** = catId;  
 **this**.**goodType** = goodType;  
}

1. onPreExecute方法，显示下载时的提示框

@Override  
**protected void** onPreExecute() {  
 **mdialog** = **null**;  
 **if**(**actionType** == ActionType.***ACTION\_DOWNLOAD***){  
 **mdialog** = **new** ProgressDialog(**mContext**);  
 **mdialog**.setTitle(D.NewGood.***HINT\_DOWNLOAD\_TITLE***);  
 **mdialog**.setMessage(D.NewGood.***HINT\_DOWNLOADING***);  
 **mdialog**.show();  
 }  
}

1. doInBackground方法，下载新品或者分类数据

@Override  
**protected** ArrayList<NewGoodBean> doInBackground(Integer... params) {  
 **int** pageId=params[0];  
 **int** pageSize=params[1];  
 Log.*i*(**TAG**,**"DownloadGoodsTask..."**);  
 ArrayList<NewGoodBean> goodList=**null**;  
 **try**{  
 **switch** (**goodType**){  
 **case** I.***NEW\_GOOD***:  
 goodList = NetUtil.*findNewandBoutiqueGoods*(**catId**,pageId,pageSize);  
 **break**;  
 **case** I.***CATEGORY\_GOOD***:  
 goodList = NetUtil.*findGoodsDetails*(**mContext**,**catId**,pageId,pageSize);  
 **break**;  
 }  
 **this**.**loadStatus** = LoadStatus.***SUCCESS***;  
 }**catch** (Exception e){  
 **this**.**loadStatus** = LoadStatus.***FAILURE***;  
 }  
 **return** goodList;  
}

1. onPostExecute方法，下载结束后，根据下载结果处理提示框和adapter

@Override  
**protected void** onPostExecute(ArrayList<NewGoodBean> goods) {  
 **if**(**loadStatus** == LoadStatus.***FAILURE***){  
 **if**(**mdialog**!=**null**){  
 **mdialog**.dismiss();  
 }  
 Toast.*makeText*(**mContext**,D.NewGood.***HINT\_DOWNLOAD\_FAILURE***,Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 **switch** (**actionType**){  
 **case *ACTION\_DOWNLOAD***:  
 **mdialog**.dismiss();  
 **mAdapter**.initItems(goods);  
 **break**;  
 **case *ACTION\_PULL\_DOWN***:  
 **mAdapter**.initItems(goods);  
 **break**;  
 **case *ACTION\_SCROLL***:  
 **if**(goods!=**null**){  
 **mAdapter**.addItems(goods);  
 **mAdapter**.setMore(**true**);  
 } **else** {  
 **mAdapter**.setMore(**false**);  
 }  
 **break**;  
 }  
}

1. 在NewGoodFragment中添加属性成员

DownloadGoodsTask **mDownloadGoodsTask**;

并在onCreateView中初始化

**mDownloadGoodsTask** = **new** DownloadGoodsTask(**mContext**,**mAdaper**,**mGoodList**,  
 I.ActionType.***ACTION\_DOWNLOAD***,0,I.***NEW\_GOOD***);  
**mDownloadGoodsTask**.execute(**mPageId**,**PAGE\_SIZE**);

1. 设置上拉刷新事件监听

**private void** setPullUpRefreshListener() {  
 mgvNewGood.setOnScrollListener(**new** OnScrollListener() {  
 **int lastPosition**;  
 @Override  
 **public void** onScrollStateChanged(AbsListView view, **int** scrollState) {  
 **if**(scrollState==OnScrollListener.***SCROLL\_STATE\_IDLE*** && **lastPosition**==**mAdaper**.getCount()-1 && **mAdaper**.isMore()){  
 **mPageId**=**mPageId**+**PAGE\_SIZE**;  
 **mDownloadGoodsTask**=**new** DownloadGoodsTask(**mContext**,**mAdaper**,  
 **mGoodList**, ActionType.***ACTION\_SCROLL***,I.***CAT\_ID***,I.***NEW\_GOOD***);  
 **mDownloadGoodsTask**.execute(**mPageId**,**PAGE\_SIZE**);  
 }  
 }  
  
 @Override  
 **public void** onScroll(AbsListView view, **int** firstVisibleItem,   
**int** visibleItemCount, **int** totalItemCount) {  
 **lastPosition**=firstVisibleItem+visibleItemCount-1;  
 }  
 });  
 }

1. 设置下拉刷新事件监听

**private void** setPullDownRefreshListener() {  
 **mprfvNewGood**.setOnRefreshListener(**new** OnRefreshListener() {  
 @Override  
 **public void** loadData() {  
 **mPageId**=0;  
 **mDownloadGoodsTask**=**new** DownloadGoodsTask(**mContext**,**mAdaper**,  
 **mGoodList**,ActionType.***ACTION\_PULL\_DOWN***,I.***CAT\_ID***,I.***NEW\_GOOD***);  
 **mDownloadGoodsTask**.execute(**mPageId**,**PAGE\_SIZE**);  
 }  
  
 @Override  
 **public** PullRefreshView.LoadStatus getLoadStatus() {  
 **return mDownloadGoodsTask**.getLoadStatus();  
 }  
 },**mgvNewGood**);  
}

# 步骤8

1. 在MainActivity中添加属性

private NewGoodFragment mNewGoodFragment;

Drawable drawableNewGood,drawableBoutique,drawableCategory,

drawableCart,drawablePersonalCenter;

1. 在initFragment中初始化，并显示第一个Fragment

mNewGoodFragment = new NewGoodFragment();

mFragments = new Fragment[] {mNewGoodFragment};

**if**(**currentTabIndex** == -1){  
 **currentTabIndex** = 0;  
 **index** = 0;  
 **drawableNewGood** = getmDrawable(R.drawable.***menu\_item\_new\_good\_selected***);  
 **mivNewGood**.setImageDrawable(**drawableNewGood**);  
 FragmentUtils.*startFragment*(**mContext**, **mFragments**[0]);  
}

1. 设置菜单项单击事件监听

MenuItemClickListener listener=**new** MenuItemClickListener();  
**mLayoutNewGood**.setOnClickListener(listener);

1. 底部菜单单击事件

**class** MenuItemClickListener **implements** View.OnClickListener {  
 @Override  
 **public void** onClick(View v) {  
 setMenuItemDefaultDrawable();  
 Fragment fragment = **null**;  
 **switch** (v.getId()) {  
 **case** R.id.***layout\_new\_good***:  
 **index** = 0;  
 **drawableNewGood** = getmDrawable(R.drawable.***menu\_item\_new\_good\_selected***);  
 fragment = **mNewGoodFragment**;  
 **break**;  
 **case** R.id.***layout\_boutique***:  
 **index** = 1;  
 **drawableBoutique** = getmDrawable(R.drawable.***boutique\_selected***);  
 **break**;  
 **case** R.id.***layout\_category***:  
 **index**=2;  
 **drawableCategory** = getmDrawable(R.drawable.***menu\_item\_category\_selected***);  
 **break**;  
 **case** R.id.***layout\_cart***:  
 **index** = 3;  
 **drawableCart** = getmDrawable(R.drawable.***menu\_item\_cart\_selected***);  
 **break**;  
 **case** R.id.***layout\_personal\_center***:  
 **index** = 4;  
 **drawablePersonalCenter** = getmDrawable(R.drawable.***menu\_item\_personal\_center\_selected***);  
 **break**;  
 }  
 **if** (**currentTabIndex** != **index**) {  
 FragmentTransaction trx = getSupportFragmentManager().beginTransaction();  
 trx.hide(**mFragments**[**currentTabIndex**]);  
 **if** (!**mFragments**[**index**].isAdded()) {  
 trx.add(R.id.***fragment\_container***, **mFragments**[**index**]);  
 }  
 trx.show(**mFragments**[**index**]).commit();  
 }  
 Log.*e*(***TAG***,**"MenuItemClickListener.currentTabIndex="**+**currentTabIndex**);**currentTabIndex** = **index**;  
 setMenuItemDrawable();  
 }  
 }

1. 添加setMenuItemDefaultDrawable方法，设置菜单项按钮顶部缺省显示的图片

**private void** setMenuItemDefaultDrawable() {  
 **drawableNewGood** = getmDrawable(R.drawable.***menu\_item\_new\_good\_normal***);  
 **drawableBoutique** = getmDrawable(R.drawable.***boutique\_normal***);  
 **drawableCategory** = getmDrawable(R.drawable.***menu\_item\_category\_normal***);  
 **drawableCart** = getmDrawable(R.drawable.***menu\_item\_cart\_normal***);  
 **drawablePersonalCenter** = getmDrawable(R.drawable.***menu\_item\_personal\_center\_normal***);  
}

1. 添加setMenuItemDrawable方法，设置菜单项按钮顶部某菜单项被选择后显示的图片

**private void** setMenuItemDrawable() {  
 **int** width = (**int**) (**mDensity** \* 32);  
 **int** height = width;  
  
 Rect bounds = **new** Rect(0, 0, width, height);  
 **drawableNewGood**.setBounds(bounds);  
 **mivNewGood**.setImageDrawable(**drawableNewGood**);  
}

1. 添加getmDrawable方法，统一获取资源的接口

**public** Drawable getmDrawable(**int** id){  
 Resources res = getResources();  
 Drawable drawable = res.getDrawable(id);  
 **return** drawable;  
}

1. 创建FragmentUtils文件，汇总启动Fragment的方法

**public static void** startFragment(FragmentActivity context, Fragment fragment){  
 FragmentManager manager = context.getSupportFragmentManager();  
 FragmentTransaction ft = manager.beginTransaction();  
 ft.replace(R.id.***fragment\_container***,fragment).commit();  
}

1. 在MainActivity中重写父类的onStart方法，添加setListener方法

# 步骤9

1. 创建商品详情类GoodDetailsActivity，继承BaseActivity，在AndroidManifest中注册。

<**activity  
 android:name="cn.ucai.fulicenter.activity.GoodDetailsActivity"  
 android:screenOrientation="portrait"  
 android:theme="@style/horizontal\_slide"**>  
</**activity**>

1. 导入图片切换类FlowIndicator，在attrs.xml中添加数据

<declare-styleable name="FlowIndicator">

<attr name="count" format="integer"/>

<attr name="r" format="dimension"/>

<attr name="space" format="dimension"/>

<attr name="normal\_color" format="color|reference"/>

<attr name="focus\_color" format="color|reference"/>

<attr name="focus" format="integer"/>

</declare-styleable>

1. 在工具类Utils中添加像素和dp互相转换的方法

public static int px2dp(Context context,int px){

int density = (int) context.getResources().getDisplayMetrics().density;

return px/density;

}

public static int dp2px(Context context,int dp){

int density = (int) context.getResources().getDisplayMetrics().density;

return dp\*density;

}

1. 导入图片轮播类SlideAutoLoopView
2. 创建商品详情布局文件activity\_good\_details.xml
3. 在GoodDetailsActivity中创建DownloadGoodDetailsTask下载商品详情信息的任务
4. 创建接收商品详情下载的广播GoodDetailsUpdateReceiver
5. 添加广播变量GoodDetailsUpdateReceiver mReceiver，添加注册广播的方法registerGoodDetailsUpdateReceiver并在onCreate中调用

private void registerGoodDetailsUpdateReceiver() {

mReceiver=new GoodDetailsUpdateReceiver();

IntentFilter filter=new IntentFilter("good\_details\_update");

registerReceiver(mReceiver, filter);

}

1. 创建ViewHolder类，封装显示商品信息的控件
2. 在GoodDetailsActivity中添加属性

GoodDetailsActivity mContext;

GoodDetailsBean mGoodDetails;

int mGoodsId;

/\*\* 用于收藏、支付的商品信息实体\*/

NewGoodBean mGood;

/\*\* 封装了显示商品信息的view\*/

ViewHolder mHolder;

SlideAutoLoopView mSlideAutoLoopView;

FlowIndicator mFlowIndicator;

/\*\* 显示颜色的容器布局\*/

LinearLayout mLayoutColors;

/\*\* 添加收藏\*/

final static int ACTION\_ADD\_COLLECT=1;

/\*\* 取消收藏\*/

final static int ACTION\_DELETE\_COLLECT=2;

/\*\* 收藏的操作码\*/

boolean misCollect;

ImageView mivCollect;

ImageView mivAddCart;

TextView mtvCartCount;

GoodDetailsUpdateReceiver mReceiver;

CartChangedReceiver mCartChangedReceiver;

ArrayList<CartBean> mCartList;

1. 在onCreate中设置布局，setContentView(R.layout.activity\_good\_details);
2. 添加初始化控件方法initView
3. 添加初始化加载数据的方法initData

**private void** initData() {  
 **mGoodsId**=getIntent().getIntExtra(D.GoodDetails.***KEY\_GOODS\_ID***, 0);  
 **new** DownloadGoodDetailsTask(**mContext**, **mGoodsId**).execute();  
}

1. 在onCreate中添加initView()和initData()方法
2. 创建购物车更新任务类UpdateCartTask
3. 创建购物车加载任务类UploadCartTask
4. 在Utils类中添加购物车相关操作
5. 添加获得购物车数量的方法initCartData，并在onCreate中调用

**private void** initCartData(){  
 **mCartList**=FuLiCenterApplication.*getInstance*().getCartList();  
 Log.*e*(***TAG***,**"CartChangedReceiver.mCartList="**+**mCartList**);  
 **if**(!**mCartList**.isEmpty()){  
 **mtvCartCount**.setVisibility(View.***VISIBLE***);  
 **int** count=Utils.*sumCartCount*();  
 Log.*e*(***TAG***,**"CartChangedReceiver.count="**+count);  
 **mtvCartCount**.setText(**""**+count);  
 }  
}

1. 添加购物车数据改变的广播CartChangedReceiver

**class** CartChangedReceiver **extends** BroadcastReceiver{  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 initCartData();  
 }  
}

1. 添加变量CartChangedReceiver mCartChangedReceiver，添加注册广播的方法registerCartChangedReceiver并在onCreate中调用

private void registerCartChangedReceiver() {

mCartChangedReceiver=new CartChangedReceiver();

IntentFilter filter=new IntentFilter("cartChanged");

registerReceiver(mCartChangedReceiver, filter);

}

1. 在GoodDetailsActivity中添加收藏商品的任务类CollectTask
2. 在GoodDetailsActivity的onCreate中添加setListener方法

private void setListener() {

setCollectClickListener();

setReturnClickListener();

setAddCartClickListener();

}

1. 收藏商品的点击事件

**private void** setCollectClickListener() {  
 **mivCollect**.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 String userName=FuLiCenterApplication.*getInstance*().getUserName();  
 **if**(userName==**null**){  
 Utils.*showToast*(**mContext**, **"请先登陆"**, Toast.***LENGTH\_LONG***);  
 **return** ;  
 }  
 **new** CollectTask(**mContext**,userName,**mGoodDetails**,**mivCollect**,**misCollect**).execute();  
 }  
 });  
}

1. 返回按钮的点击事件

**private void** setReturnClickListener() {  
 findViewById(R.id.***ivReturn***).setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 finish();  
 }  
 });  
}

1. 添加商品的点击事件

**private void** setAddCartClickListener() {  
 **mivAddCart**.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Utils.*addCart*(**mContext**,**mGoodDetails**);  
 }  
 });  
}

1. 重写父类的onDestroy方法，注销广播

protected void onDestroy() {

super.onDestroy();

if(mReceiver!=null){

unregisterReceiver(mReceiver);

}

if(mCartChangedReceiver!=null){

unregisterReceiver(mCartChangedReceiver);

}

}

# 步骤10

1. 创建DownloadCartTask任务类
2. 在MainActivity中添加购物车数据改变的广播

class CartChangedReceiver extends BroadcastReceiver{

@Override

public void onReceive(Context context, Intent intent) {

//统计购物车中的商品件数

ArrayList<CartBean> cartList =

FuLiCenterApplication.getInstance().getCartList();

int count=0;

for(int i=0;i<cartList.size();i++){

count+=cartList.get(i).getCount();

}

//显示购物车中的商品件数

mtvCartHint.setText(""+count);

mtvCartHint.setVisibility(View.VISIBLE);

}

}

1. 添加变量CartChangedReceiver mCartChangedReceiver，添加注册广播的方法并在onResume中调用

private void registerCartChangedReceiver() {

mCartChangedReceiver=new CartChangedReceiver();

IntentFilter filter=new IntentFilter("cartChanged");

registerReceiver(mCartChangedReceiver, filter);

}

1. 在onDestroy中注销该广播
2. 在onCreate中调用DownloadCartTask任务类执行

**new** DownloadCartTask(**mContext**, I.***PAGE\_ID\_DEFAULT***,I.***PAGE\_SIZE\_DEFAULT***).execute();

# 步骤11

1. 创建CartFragment文件并继承Fragment，并重写onCreateView方法
2. 创建fragment\_cart.xml布局文件