中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师： 郑贵峰

年级	15 级	专业（方向）	软件工程（移动信息工程） 互联网方向
学号	15352211	姓名	林苗
电话	13763360840	Email	554562948@qq.com
开始日期	2017.10.30	完成日期	2017.10.31

一、 实验题目

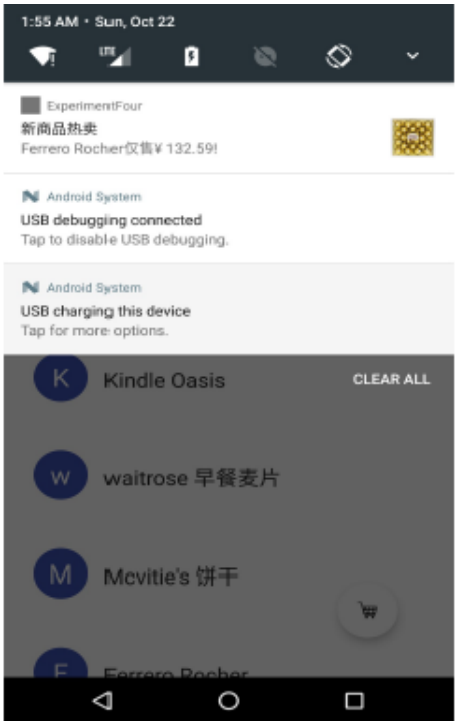
实验四 Broadcast 使用

二、 实现内容

在实验三的基础上，实现静态广播、动态广播两种改变 Notification 内容的方法。

具体要求：

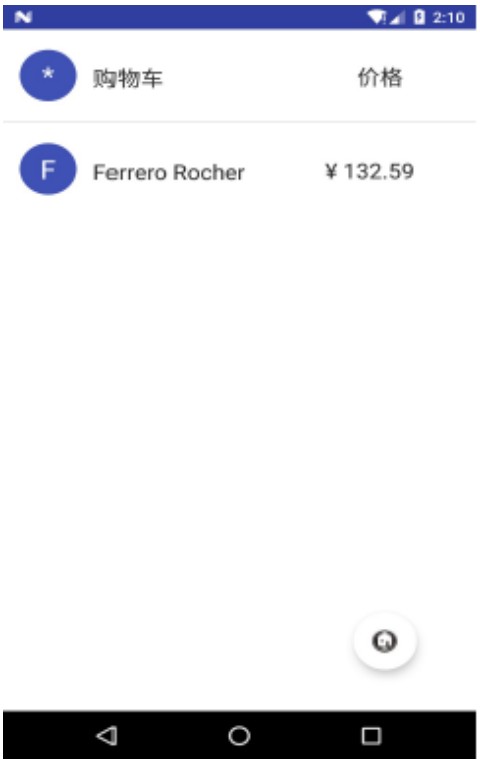
(1)在启动应用时，会有通知产生，随机推荐一个商品:



(2)点击通知跳转到该商品详情界面:

(3)点击购物车图标，会有对应通知产生，并通过 Eventbus 在购物车列表更新数据:

(4) 点击通知返回购物车列表:

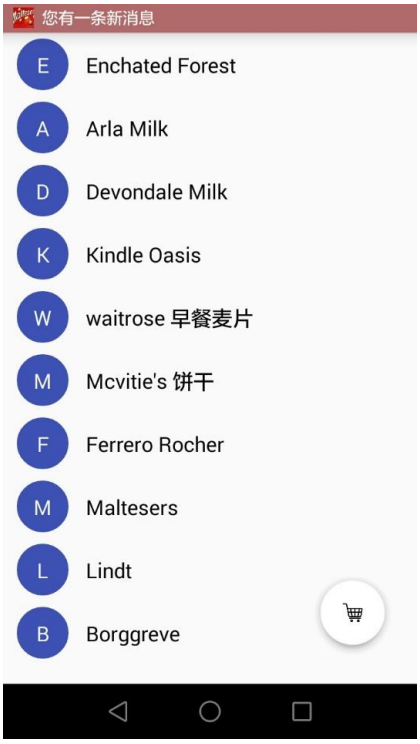


(5)实现方式要求:启动页面的通知由静态广播产生，点击购物车图标的通知由动态广播产生。

三、 课堂实验结果

（1） 实验截图

打开 app 界面，通知栏会显示“您有一条新消息”：

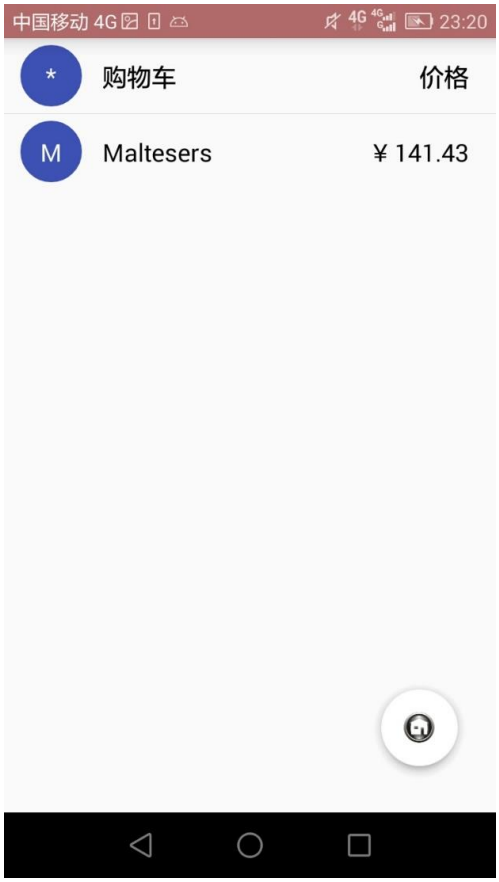


通知栏向下滑动，看到商品推荐：

点击商品推荐，弹出商品详情界面：

点击购物车图标，通知栏显示马上下单：

单击马上下单，跳转到购物车界面，新商品已被加入购物车：



（2） 实验步骤及关键代码

- 静态广播实现商品推荐

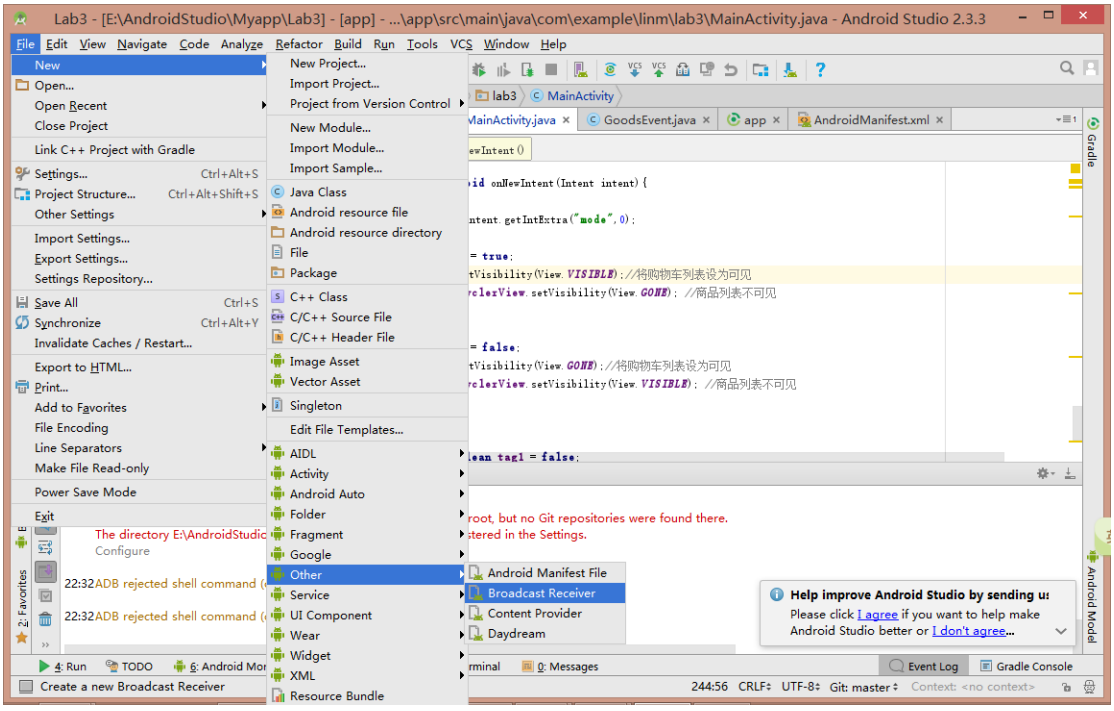
在 MainActivity 中产生随机数：

```
Random random = new Random();
```

传进一个随机数，发送广播：

```
//静态广播
Intent intent = new Intent();
intent.setAction("StaticBroadcast");
intent.putExtra("info", random.nextInt(Infos.size()));
sendBroadcast(intent);
```

如图方法，创建 MyReceiver 的 java 类



在 AndroidManifest.xml 中，修改静态广播的名字为 StaticBroadcast：

```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="StaticBroadcast" />
    </intent-filter>
</receiver>
```

在 MyReceiver.java 中，重写 MyReceiver 的 onReceive 函数：

```
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
```

获得随机数：

```
int i = intent.getIntExtra("info", 0);
```

设置状态栏提示 Notification：

```
Notification.Builder builder = new Notification.Builder(context);
builder.setContentTitle("新商品热卖") //标题
        .setContentText(MainActivity.Infos.get(i).getName()+"仅售"+MainActivity.Infos.get(i).getPrice()+"!") //内容
        .setAutoCancel(true)
        .setTicker("您有一条新消息") //提示消息
        .setSmallIcon(MainActivity.Infos.get(i).getImgid()) //小图标
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), MainActivity.Infos.get(i).getImgid())); //大图标
```

用 PendingIntent、setContentIntent 设置点击跳转：

```
//跳转到商品详情介绍页面
Intent x= new Intent(context, GoodsInfoActivity.class);
Bundle bundle = new Bundle();
String productname = MainActivity.Infos.get(i).getName();
bundle.putString("name",productname);//属性为name，数据为productname
x.putExtras(bundle);
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
        x, PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(pendingIntent);
```

显示 Notification：

```
Notification notify = builder.build();
//manager负责将状态显示出来
NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
int uniqueId = (int) System.currentTimeMillis(); //设置随机数，不会覆盖之前的通知
manager.notify(uniqueId,notify);
```

- 动态广播提示通知商品已加入购物车

创建新类 DynamicReceiver，重写 onReceive 函数：

```
class DynamicReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
```

获取图片 id：

```
int Imgid = intent.getIntExtra("imgid",0);
```

设置 Notification 显示的内容：

```
Notification.Builder builder = new Notification.Builder(context);
//设置Notification显示内容
builder.setContentTitle("马上下单")
        .setContentText(intent.getStringExtra("name")+"已经添加到购物车")
        .setSmallIcon(Imgid)
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(),Imgid))
        .setAutoCancel(true);
```

设置跳转到购物车界面：

```
//跳转到购物车界面
Intent x= new Intent(context, MainActivity.class);
x.putExtra("mode",1); //传值
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
        x, PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(pendingIntent);
```

执行 Notification，与静态广播类似：

```
//执行
Notification notify = builder.build();
NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
int uniqueId = (int) System.currentTimeMillis();
manager.notify(uniqueId,notify);
```

在商品详情界面 GoodInfoActivity.java 中，声明 DynamicReceiver 的类的实例：

```
DynamicReceiver dynamicReceiver;
```

在 onCreate 函数中注册广播接收：

```
IntentFilter filter = new IntentFilter();
filter.addAction("DynamicBroadcast");
dynamicReceiver = new DynamicReceiver();
//注册广播接收
registerReceiver(dynamicReceiver,filter);
```

重写 onDestroy 函数：

```
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(dynamicReceiver);
}
```

在 onCreate 函数中的购物车按钮重写的 onClick 函数中，加上发送动态广播的代码：

```
Intent intent = new Intent();
intent.setAction("DynamicBroadcast");
intent.putExtra("name", p.getName());
intent.putExtra("imgid", p.getImgid());
sendBroadcast(intent);
```

在 MainActivity.java 中，重写 onNewIntent 函数，使之跳转到购物车界面：

```
@Override
protected void onNewIntent(Intent intent) {
    final FloatingActionButton convert = (FloatingActionButton) findViewById(R.id.convert);
    int y = intent.getIntExtra("mode", 0);
    if (y==1) {
        convert.setImageResource(R.mipmap.mainpage);
        tag1 = true;
        LV.setVisibility(View.VISIBLE);//将购物车列表设为可见
        mRecyclerView.setVisibility(View.GONE); //商品列表不可见
    }
    else{
        convert.setImageResource(R.mipmap.shoplist);
        tag1 = false;
        LV.setVisibility(View.GONE);//将购物车列表设为可见
        mRecyclerView.setVisibility(View.VISIBLE); //商品列表不可见
    }

}
```

在 AndroidManifest.xml 文件中,将 MainActivity 的 lauchMode 改为 singleTask，使之不会另外新建一个购物车列表：

```
<activity android:name=".MainActivity"
    android:launchMode="singleTask">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

- EventBus 实现不同 activity 间的通信

在 gradle 中添加如下依赖：

```
compile 'org.greenrobot:eventbus:3.0.0'
```

新开一个类 GoodsEvent：

```
public class GoodsEvent{
    private Map<String, Object> goods = new LinkedHashMap<>();
    public GoodsEvent(Map<String, Object> temp){
        goods = temp;
    }
    public Map<String, Object> getGoods() { return goods; }
}
```

在 GoodsInfoActivity.java 的购物车图标单击事件 onClick 函数中，注释掉 lab3 使用的直接添加到购物车列表 shoplist 的方法，使用 EventBus 发送：

```
    public void onClick(View v) {
        Map<String, Object> temp = new LinkedHashMap<>();
        temp.put("FirstLetter", p.getFirstLetter());
        temp.put("name",p.getName());
        temp.put("price",p.getPrice());
//      MainActivity.shoplist.add(temp);
//      MainActivity.simpleListAdapter.notifyDataSetChanged();
        EventBus.getDefault().post(new GoodsEvent(temp));
```

在 MainActivity.java 的 onCreate 函数中，注册 EventBus：

```
EventBus.getDefault().register(this);
```

重写 onEventMainThread 函数，接收到商品后，添加到购物车队列，更新 Adapter：

```
@Subscribe
public void onEventMainThread(GoodsEvent event) {

    shoplist.add(event.getGoods());
    simpleListAdapter.notifyDataSetChanged();
}
```

注销 EventBus：

```
@Override
protected void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().unregister(this);
}
```

（3） 实验遇到的困难及解决思路

- 用 random 产生整型数，但是在 activity 中显示 null。
  解决思路：观察到之前的设置是输出 getStringExtra(“info”)，接收到的类型不是整数，查了一下资料，intent 传递整型数的写法如下：

发送方：

```
Intent intent = new Intent();
intent.putExtra("ID", arg);
startActivity(intent);
```

其中，arg 为一个 int 型变量。

接收方：

```
Intent intent = getIntent();
int id = intent.getIntExtra("ID",0);
```

- 用大量的 if-else 语句根据不同的商品名字设置对应的图片，过程很繁琐，代码也不整洁。

  解决思路：重写商品的类，将原本最后一项的字符串 background 改为 int 型变量：

```
public class Info implements Serializable {

    private String name; //名字
    private String price; // 价格
    private String type; //类型
    private String info; //信息
    private int imgid;//背景
```

getImgid()函数：

```
public int getImgid() { return imgid; }
```

其中 imgid 直接传入对应的背景图标，例如在 MainActivity 中初始化商品队列 Infos：

```
add(new Info("Enchated Forest", "¥ 5.00", "作者", "Johanna Basford", R.mipmap.enchatedforest));
```

在用 getImgid()可获得对应的图片 id，是 int 型的：

```
builder.setContentTitle("新商品热卖") //标题
        .setContentText(MainActivity.Infos.get(i).getName()+"仅售"+MainActivity.Infos.get(i).getPrice()+"!") //内容
        .setAutoCancel(true)
        .setTicker("您有一条新消息") //提示消息
        .setSmallIcon(MainActivity.Infos.get(i).getImgid()) //小图标
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), MainActivity.Infos.get(i).getImgid())); //大图标
```

- 一开始在发送方和接收方都注册和注销 EventBus，执行到相应动作时会闪退。

  解决思路：通过排除法，先注释掉发送方，即 GoodsInfoActivity.java 中的注册和注销 EventBus 部分的代码，发现不会出现闪退的问题。故只需要在接收方注册和注销 EventBus，不需要在发送方注册，发送方直接使用：

```
EventBus.getDefault().post(new GoodsEvent(temp) );
```

发送即可，其中 GoodsEvent 为事件类，temp 为其构造函数传入的值。

接收方需要重写 onEventMainThread()函数：

```
@Subscribe
public void onEventMainThread(GoodsEvent event) {

    //接收到event之后要执行的动作

}
```

- 打开 app，点开任意一个商品详情介绍页面，点击通知栏中出现的商品推荐，出现另一个商品详情介绍界面，可以正常返回到上一个商品详情介绍页面，再一次返回就会闪退。

  解决思路：注销动态广播的函数有问题，原先使用：

```
protected void onPause() {
    super.onPause();
    unregisterReceiver(dynamicReceiver);
}
```

  更改为：

```
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(dynamicReceiver);
}
```

  要使用原先有的 onDestroy()函数。

四、 实验思考及感想

本次实验改进了传递图片的写法，在商品类加上一个属性 imgid，直接使用对应图片的数值（如 imgid = R.mipmap.arla）。setImageResource()接收的参数为 int 型，在设置图片的时候，直接用 getImgid()得到对应商品类的图片 int 数值，传入 setImageResource()即可设置对应图片。这种写法免去了大量的 if-else 语句，使代码更加简洁。

由于 bug 比较多，需要慢慢调试，在网上找到打印的方法，在需要的地方加上：

```
String Name;
Log.e("name",Name);
```

打印的消息可以在 Logcat 中看到（6：Android Monitor）。

本例中，Name 是一个 String 变量，当程序执行到该位置时，就会打印“name”+Name 字符串。是一种很好的 debug 的方法。

参考资料：

Android 中 BroadcastReceiver 的两种注册方式（静态和动态）详解
http://blog.csdn.net/panhouye/article/details/53588930
Android Notification 常见样式总结
http://blog.csdn.net/w804518214/article/details/51231946

Android 中的广播 Broadcast 详解
http://blog.csdn.net/jiangwei0910410003/article/details/19150705
Notification 简单实例
http://blog.csdn.net/guchuanhang/article/details/51601303
EventBus 使用详解(一)——初步使用 EventBus
http://blog.csdn.net/harvic880925/article/details/40660137