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：郑贵锋

年级	15 级	专业（方向）	软件工程(移动)
学号	15352133	姓名	黄少豪
电话	13727024545	Email	328730316@qq.com
开始日期	2017.10.28	完成日期	2017.10.30

一、 实验题目

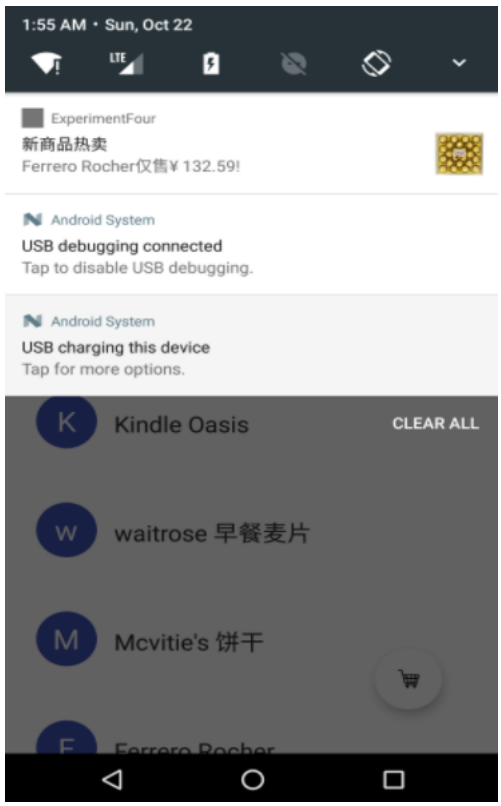
Broadcast 使用

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。

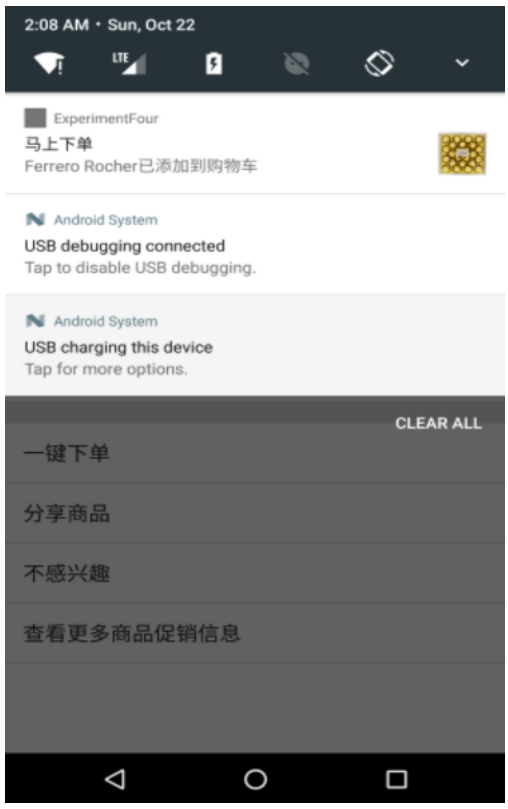
具体要求：

(1) 在启动应用时，会有通知产生，随机推荐一个商品：

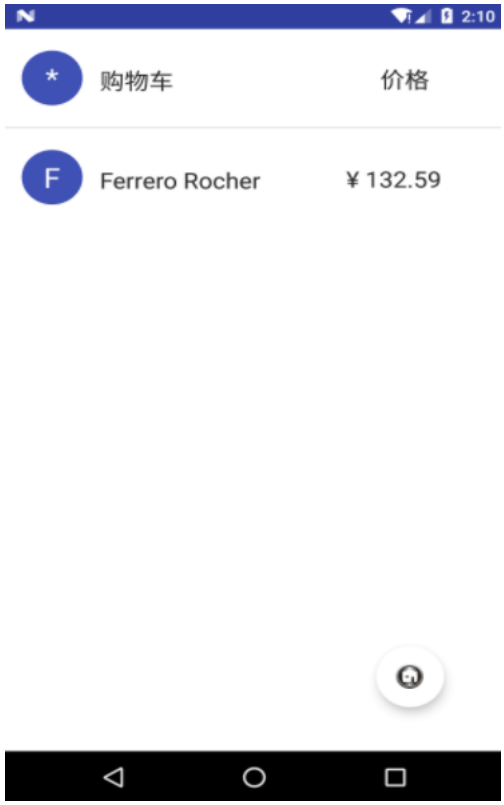


(2) 点击通知跳转到该商品详情界面:

(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:

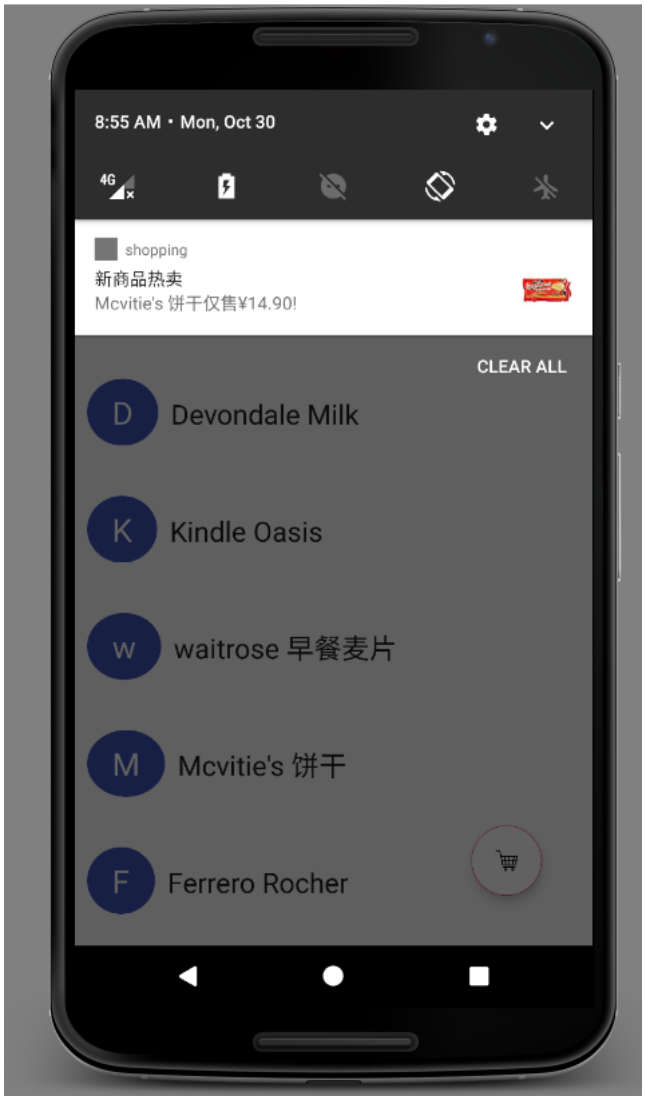


(5) 实现方式要求:启动页面的通知由静态广播产生，点击购物车图标的通知由动态广播产生。

三、 课堂实验结果

（1） 实验截图

Android Emulator - Nexus_6_API_25:5554




Android Emulator - Nexus_6_API_25:5554
8:56
Digestive
Mcvitie's 饼干
¥14.90
产地 英国
更多产品信息
一键下单
分享商品
不感兴趣
查看更多产品促销信息


Android Emulator - Nexus_6_API_25:5554
8:57 AM • Mon, Oct 30
shopping
马上下单
Mcvitie's 饼干已添加到购物车
Mcvitie's 饼干
CLEAR ALL
¥14.90
产地 英国
更多产品信息
一键下单
分享商品
不感兴趣
商品已添加到购物车

（2） 实验步骤以及关键代码

1. 自定义一个静态广播接收类

```
public class StaticReceiver extends BroadcastReceiver{
    Notification.Builder builder;
    Notification notify;
    @Override
    public void onReceive(Context context, Intent intent){
        if (intent.getAction().equals("STATICATION")){
            Bundle bundle = intent.getExtras();
            int pos = bundle.getInt("pos");
            Bitmap bitmap = BitmapFactory.decodeResource(context.getResources(), MainActivity.pic.get(pos));
            Intent mainIntent = new Intent(context, InfoActivity.class);
            mainIntent.putExtra("pos", pos);
            PendingIntent mainPendingIntent = PendingIntent.getActivity(context, 0, mainIntent, PendingIntent.FLAG_UPDATE_CURRENT);
```

变量 pos 表示当前商品是商品列表第 pos 项商品；通过 bitmap 变量可获取存储在 MainActivity.pic 列表里的第 pos 项商品图片；PendingIntent 对象用来存储点击通知后发生的 Intent 事件，这里当点击通知后会跳转到第 pos 项商品的详情界面(InfoActivity)并且会传递

一个 pos 数据过去表示正在处理第 pos 项商品。

```
builder = new Notification.Builder(context);
builder.setContentTitle("新商品热卖")
        .setContentText(MainActivity.listItems.get(pos).get("name").toString() + "仅售"
                + MainActivity.listItems.get(pos).get("price").toString() + "!")
        .setTicker("您有一条新消息")
        .setSmallIcon(MainActivity.pic.get(pos))
        .setLargeIcon(bitmap)
        .setAutoCancel(true)
        .setContentIntent(mainPendingIntent);
notify = builder.build();
MainActivity.manager.notify(0, notify);
```

接着通过 Notification.Builder 函数创建 Builder 对象并通过这个 Builder 对象设置通知的样式，包括标题，内容，小图标，大图标以及点击事件。其中大图标使用前面定义的 Bitmap 对象，点击事件使用前面定义的 PendingIntent 对象。Builder 对象创建完成后就可以调用其 build 方法创建一个通知，然后通过通知管理器(NotificationManager)对象的 notify 方法使通知显现。

2. 在 app 启动后，应用发出一个广播，使应用产生一个随机商品推荐通知。

```
<receiver android:name=".StaticReceiver"
    android:exported="true">
    <intent-filter>
        <action android:name="STATICATION"/>
    </intent-filter>
</receiver>
```

先在 AndroidManifest.xml 文件中静态注册一个广播接收器，广播接收器类型采用前面自定义的 StaticReceiver 类；action 属性的 name 标记为 STATICATION 表示只接收类型为 STATICATION 类型的广播。

```
public void initBroadcast(){
    Random random = new Random();
    int pos = random.nextInt(10) + 1;
    Bundle bundle = new Bundle();
    bundle.putInt("pos", pos);
    Intent intentBroadcast = new Intent("STATICATION");
    intentBroadcast.putExtras(bundle);
    sendBroadcast(intentBroadcast);
}
```

通过 Random 的 nextInt 方法随机生成一个 1 到 10 范围内的随机整数数；新建一个 Intent 表示 STATICATION 类型的广播；bundle 存储广播的数据，这里将商品在商品列表的位置 pos 作为数据广播；最后

调用 sendBroadcast 方法发送这个广播。

这样应用就会存在一个 StaticReceiver 对象等待接收这个类型为 STATICATION 的广播，当接收到这个广播后会调用里面的 onReceive 方法。

3. 自定义一个动态广播接收类

```
public class DynamicReceiver extends BroadcastReceiver{
    Notification.Builder builder;
    Notification notify;
    @Override
    public void onReceive(Context context, Intent intent){
        if (intent.getAction().equals("DYNAMICATION")){
            Bundle bundle = intent.getExtras();
            int pos = bundle.getInt("pos");
            Bitmap bitmap = BitmapFactory.decodeResource(context.getResources(),
                    MainActivity.pic.get(pos));
            Intent mainIntent = new Intent(context, MainActivity.class);
            PendingIntent mainPendingIntent = PendingIntent.getActivity(context, 0,
                    mainIntent, PendingIntent.FLAG_UPDATE_CURRENT);
            builder = new Notification.Builder(context);
            builder.setContentTitle("马上下单")
                    .setContentText(MainActivity.listItems.get(pos).get("name").toString() + "已添加到购物车")
                    .setTicker("您有一条新消息")
                    .setSmallIcon(MainActivity.pic.get(pos))
                    .setLargeIcon(bitmap)
                    .setAutoCancel(true)
                    .setContentIntent(mainPendingIntent);
            notify = builder.build();
            MainActivity.manager.notify(0, notify);
        }
    }
```

代码几乎与 StaticReceiver 类的定义一致，只是通知的标题与内容的文本有些不同；接收的广播类型为 DYNAMICATION；PendingIntent 对象存储的点击事件没有数据传送，只是普通的界面跳转，点击通知后应用会跳转至购物车界面(MainActivity)。

4. 点击购物车后发送一个类型为 DYNAMICATION 的广播。

```
Bundle bundle = new Bundle();
bundle.putInt("pos", pos);
Intent intentBroadcast = new Intent("DYNAMICATION");
intentBroadcast.putExtras(bundle);
sendBroadcast(intentBroadcast);
```

变量 pos 表示当前处理的是第 pos 项商品，这个变量作为数据传递至通知接收器。

5. 动态注册一个只接收 DYNAMICATION 类型的广播接收器。

```
dynamicReceiver = new DynamicReceiver();
dynamic_filter = new IntentFilter();
dynamic_filter.addAction("DYNAMICATION");
registerReceiver(dynamicReceiver, dynamic_filter);
```

变量 dynamicReceiver 是一个 DynamicReceiver 对象；dynamic_filter 是一个广播接收过滤器对象，调用其 addAction 方法使其只会过滤 DYNAMICATION 类型的广播；调用 registerReceiver 方法使通知接收对象与通知过滤对象绑定在一起，注册一个通知接收器只处理类型为 DYNAMICATION 的广播。

6. 使用 EventBus 实现点击购物车按钮购物车列表添加商品数据的功能。

```
public class MessageEvent {
    int pos;
    public MessageEvent(int pos){this.pos = pos;}
    public int getMessage(){return pos;}
}
```

先定义一个事件类，这个事件只包含一个数据 pos，表示第 pos 项商品。

```
public void carClick(View view){
    Toast.makeText(InfoActivity.this, "商品已添加到购物车",
            Toast.LENGTH_SHORT).show();
    EventBus.getDefault().post(new MessageEvent(pos));
}
```

点击购物车按钮后，通过 EventBus 发送一个事件，这个事件包含一个 pos 数据，表示当前处理第 pos 项商品。

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

在主界面注册使用 EventBus。

```
@Subscribe
public void onMessageEvent(MessageEvent event){
    int pos = event.getMessage();
    listAdapter.goods.add(pos);
    listAdapter.notifyDataSetChanged();
    Bundle bundle = new Bundle();
    bundle.putInt("pos", pos);
    Intent intentBroadcast = new Intent("DYNAMICATION");
    intentBroadcast.putExtras(bundle);
    sendBroadcast(intentBroadcast);
}
```

在主界面注册一个事件处理函数，当有一个 MessageEvent 类型的事件发生时，这个函数会被调用。这里是取出 event 里面的 pos 数据获得当前正在处理的商品的位置 pos，然后往购物车列表添加据 pos 使

购物车列表添加商品数据。最后一段是广播发送，在前面已经提过。

7. 点击通知跳转到购物车列表，使用 onNewIntent 方法使主界面跳转至购物车列表界面。

```
@Override
protected void onNewIntent(Intent intent){
    super.onNewIntent(intent);
    setIntent(intent);
    fab = (FloatingActionButton) findViewById(R.id.fab);
    mRecyclerView = (RecyclerView) findViewById(R.id.recycler_view);
    mListView = (ListView) findViewById(R.id.list_view);
    fab.setImageResource(R.mipmap.mainpage);
    tag = false;
    mRecyclerView.setVisibility(View.GONE);
    mListView.setVisibility(View.VISIBLE);

}
```

修改 FloatingActionButton 的图标；tag 设置为 false 表示显示购物车列表；使商品列表不可见；使购物车列表可见。

（3） 实验遇到困难以及解决思路

1. 点击按钮跳转至购物车界面，设置 intent 跳转到 MainActivity 怎么都是跳转到商品列表界面而无法跳转到购物车界面。解决方法：在 MainActivity 中重写 onNewIntent 函数，在 onNewIntent 函数中将商品列表隐藏显现购物车列表，同时修改悬浮按钮图标。
2. 点击通知跳转至商品详情界面再点击返回按钮跳转到主界面，发现一些回调数据没有传回。原因是商品详情界面不是从主界面跳转过来的而是从点击通知过来的，所以主界面的回调数据处理函数没有触发。解决方法：将数据封装到事件里，使用 EventBus 发出事件与捕获事件进行数据处理。

四、 课后实验结果

五、 实验思考及感想

1. 静态注册简单但不够灵活，不能在一些特殊情况通过条件判断来决定是否进行注册；动态注册耦合性强代码不够好看但比较灵活。
2. EventBus 的使用比起数据回调使用更简单也更易理解，发布与订阅的模式也使代码结构更优美，能处理更多复杂的情况。
3. onNewIntent 函数，个人理解就是使同一个 Activity 对象再次加载接收一个外部 Intent 并且执行一系列行为。在本次实验中，MainActivity 的 lauchMode 是 singleInstance，即在栈中这个 Activity 始终只有一份。那么这里就有一个问题，当从另一个 Activity 传递一个 Intent 到 MainActivity，那么在 MainActivity 中会在哪里开始执行呢？答案就是在 onNewIntent 函数中执行。所以，个人觉得 onNewIntent 函数就像是一个订阅了传递 Intent 到该界面的事件一样，当外部有一个 Intent 传递过来就能触发 onNewIntent 函数进行处理。