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	2015	专业（方向）	互联网
学号	15352194	姓名	梁杰鑫
电话	15113959962	Email	Alcanderian@gmail.com
开始日期	2017.10.25	完成日期	2017.11.1

一、 实验题目

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。

二、 实现内容

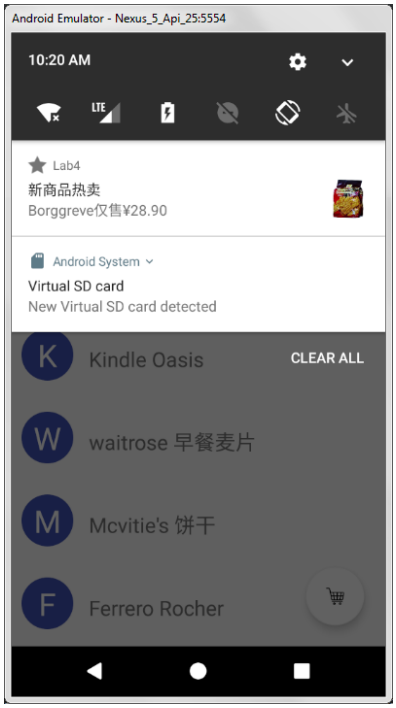
(1) 在启动应用时，会有通知产生（点击通知跳转到该商品详情界面），随机推荐一个商品

(2) 点击购物车图标，会有对应通知产生（点击通知返回购物车列表），并通过 Eventbus 在购物车列表更新数据

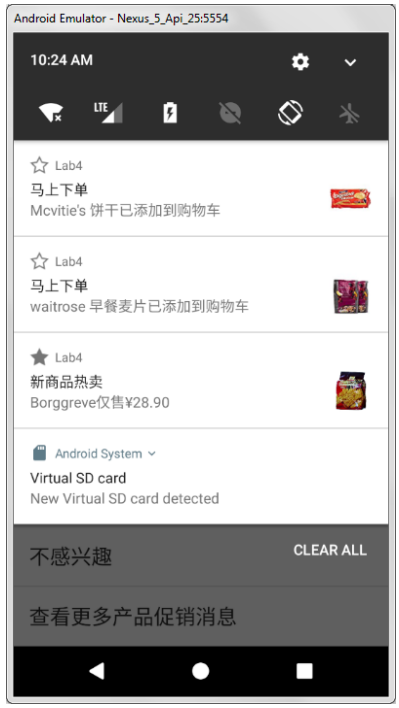
(3) 实现方式要求：启动页面的通知由静态广播产生，点击购物车图标的通知由动态广播产生。

三、 课堂实验结果

（1） 实验截图



打开 app 发放通知



商品加入购物车发放通知

点击热卖通知进入商品页面　　点击“马上下单”进入购物车

（2） 实验步骤以及关键代码

- Receiver

首先要新建一个 Receiver 类，然后再在 AndroidManifest 中注册这个 Receiver，并在 Filter 中加入 Action。Filter 的意义在于过滤广播，只有符合 Filter 中 Action 的广播才会触发这个 Receiver。

```
        <receiver android:name=".MyReceiver">
            <intent-filter>
                <action android:name="top.alau.lab4.broadcast.STATIC" />
            </intent-filter>
        </receiver>
```

在 Receiver 中，我们只需要重载 onReceive 函数即可。

```
    @Override
    public void onReceive(Context context, Intent intent) {
        MyShopApp my_app = (MyShopApp) context.getApplicationContext();
        int iid = intent.getIntExtra( name: "itemid",  defaultValue: 0);
        int idx = my_app.good_data.getIndex(iid);
        int imgid = my_app.good_data.imgid.get(idx);
        Map<String, Object> good_data = my_app.good_data.data.get(idx);
        Intent it_noti;
        String msg, title;
        int small_icon;

        if (intent.getAction().equals(context.getString(R.string.static_broadcast_action))) {
            it_noti = new Intent(context, DetailActivity.class);
            it_noti.putExtra( name: "itemid", iid);
            msg = good_data.get("name").toString() + "仅售"
                    + good_data.get("price").toString();
            title = "新商品热卖";
            small_icon = R.mipmap.full_star;
        } else {
            it_noti = new Intent(context, MainActivity.class);
            it_noti.putExtra( name: "mode",  value: 2);
            msg = good_data.get("name").toString() + "已添加到购物车";
            title = "马上下单";
            small_icon = R.mipmap.empty_star;
        }
```

在这里我将动态的 Receiver 和静态 Receiver 合二为一了，只需要判断 Action 的类型来投放不同的内容即可。下面的通知的投放：

```
PendingIntent pit_noti = PendingIntent.getActivity(context, requestCode: 0, it_noti,
        PendingIntent.FLAG_UPDATE_CURRENT);

Notification.Builder bu_noti = new Notification.Builder(context)
        .setContentIntent(pit_noti)
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), imgid))
        .setSmallIcon(small_icon)
        .setContentTitle(title)
        .setContentText(msg)
        .setTicker("您有一条新消息")
        .setWhen(System.currentTimeMillis())
        .setAutoCancel(true)
        .setOngoing(true);
Notification noti = bu_noti.build();
noti.flags |= Notification.FLAG_NO_CLEAR;
NotificationManager man_noti =
        (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
man_noti.notify(my_app.notify_id++, noti);
```

这里值得注意的地方有两个：

1. PendingIntent：这是一个 Intent 的容器，用来存放即将发生的 Intent 及其绑定的容器。通知触发时就会正式触发这个 Intent。
2. NotificationManager.notify 的第一个参数：这个参数是 notification 的 id，如果两个 Notification 的 id 相同就会互相覆盖，如果想让 Notification 不互相覆盖，就需要给每个 Notification 不同的 id。

- 动态 Reciever 的注册

  我们在需要用到这个 Receiver 时注册即可。这里是在商品页面注册，因为只有在这个页面才会发出动态 Receiver 接收的广播。

```
ft_cart = new IntentFilter();
ft_cart.addAction("top.alau.lab4.broadcast.DYNAMIC");
br_dynamic = new MyReceiver();
registerReceiver(br_dynamic, ft_cart);
```

要注意的问题：不能过早地注销 Receiver，系统可能会造成发送到 Receiver 的广播延迟。

- 发送广播

```
        sendBroadcast(new Intent()
                .setAction("top.alau.lab4.broadcast.DYNAMIC")
                .putExtra( name: "itemid", iid));
        Toast.makeText( context: DetailActivity.this
```

调用 sendBoradcast 即可。

- 重载 onNewIntent 方法

  当一个 Activity 实例化的时候可以接受一次 Intent，但是之后也有可能有新的 Intent 送过来，这时候就要重载 onNewIntent 方法来处理新的 Intent。

```
@Override
public void onNewIntent(Intent it) {
    super.onNewIntent(it);
    if(it.hasExtra( name: "mode")) {
        view_mode = getIntent().getIntExtra( name: "mode", defaultValue: 0);
        update_mode();
    }
}
```

- EventBus 的参数 Class

  EventBus 可以接受任何类型的对象（int 不是对象，但 Integer 是对象），这里我们可以自定义一个 Class 作为 EventBus 的参数。

```
public class MyEvent {
    String msg;

    public MyEvent(String s) { msg = s; }
}
```

然后我们只需要在 MainActivity 中注册 EventBus 并写一个订阅者函数去订阅 EventBus 中的内容即可。

```
    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onEvent(MyEvent e) {
        if (e.msg.equals("refresh cart")) {
            ((MyLvAdapter) lv_items.getAdapter()).notifyDataSetChanged();
        }
    }
```

当需要向 MainActivity 中发送 Event 时，我们调用 EventBus.post()即可。

```
                EventBus.getDefault().post(new MyEvent( s: "refresh cart"));
```

（3） 实验遇到困难以及解决思路

1. 第一次运行程序时，发现通知会互相覆盖。

   解决办法：NotificationManager.notify 的第一个参数：这个参数是 notification 的 id，如果两个 Notification 的 id 相同就会互相覆盖，如果想让 Notification 不互相覆盖，就需要给每个 Notification 不同的 id。

   这时候就要在自定义的 Application 类中这只一个全局变量来记录通知的 id，每次调用 notify 时传入(id++, notification)即可。

2. 在 MainActivity 已经实例化时，如何获得新的 Intent。

   重载 newIntent。

3. 动态广播在哪里注册？

   在需要用到动态广播 Receiver 时注册，不能在 MainActivity 注册，因为我们在商品详细页面才会发送动态广播，这时候 MainActivity 已经被系统销毁的话，可能动态 Receiver 就会收不到通知。

四、 思考及感想

学习了通知栏和广播的使用方法，难度并不大。但这会涉及到很多兼容性问题，比如 API26 中广播的权限有限制，可能发不出广播；小米手机不能自定义通知图标；新的 API 不会显示通知的 Ticker；新的 API 小图标不支持彩色图片，只能用单色图标等等。要实现全系统兼容是一件不容易的事。

EventBus 是一个好东西，它也相当于一个广播系统，但是 Filter 不限于 Action，可以是任意的数据类型。比如你 post 一个类型 XXX 的参数到 EventBus，那订阅这个类型参数的 Class 就会收到你的 Event。可以说非常灵活。