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：

年级	15	专业（方向）	移动互联网
学号	15352155	姓名	赖贤城
电话	13727024851	Email	754578682@qq.com
开始日期	2017/10/29	完成日期	2017/10/30

一、 实验题目

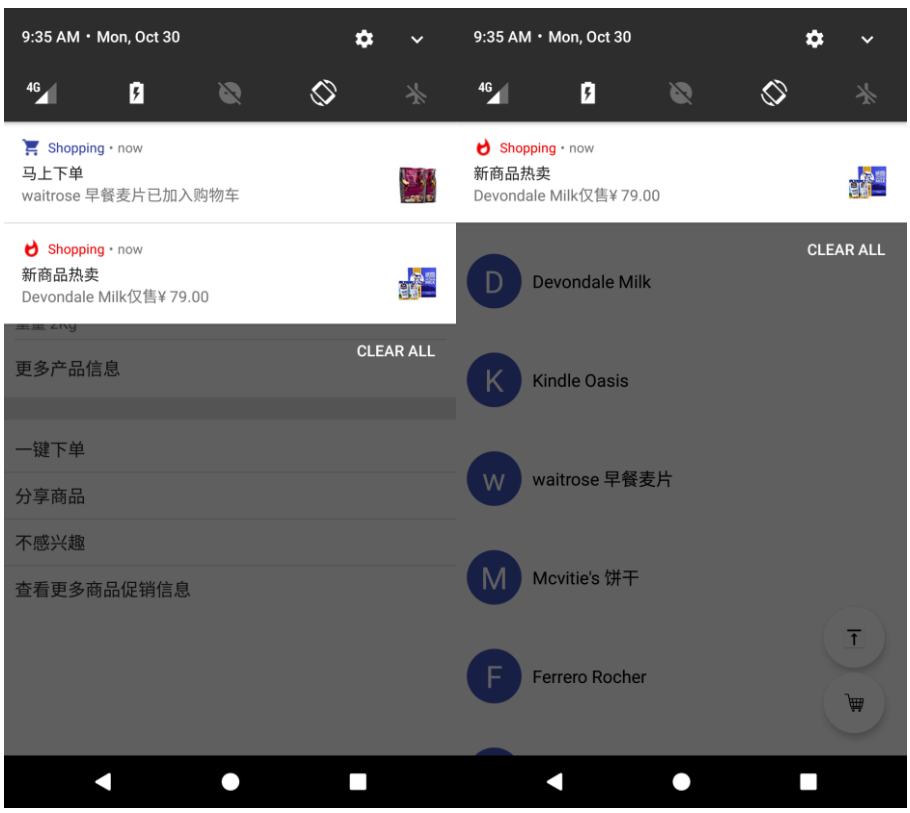
Broadcast 使用

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

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1. 进入主界面用静态广播发一条“热卖”通知

a. 主界面发送广播

在 onCreate 调用的时候发送广播，使用随机数获得一个索引，将该索引对应的商品对象包装在 intent 里面传递给广播。（注：下图函数在 onCreate 里面调用）

```
/**
 * 发送产生“热卖”通知的广播
 */
private void sendHotBroadcast() {
    Intent intent = new Intent( action: "com.lxc.my.LAUNCH");
    Random random = new Random();
    int randIndex = random.nextInt(goodsItems.size());
    intent.putExtra(LAUNCH_BROADCAST_INFO, goodsItems.get(randIndex));
    sendOrderedBroadcast(intent,  receiverPermission: null);
}
```

b. 重写 BroadcastReceiver，定义收到广播的时候执行的动作

安卓里面的 notification 采用构建者模式（builder），因此这里先生成一个 builder 实例，用它来进行各种设置（通知的标题，内容等等）。

1. 其中的 setLargeIcon 要传入一个 bitmap 对象，这里使用 BitmapFactory 里面的 decodeResouces 函数将 drawable 的资源解码成 Bitmap 对象
2. setSmallIcon 函数传入的是一个图片资源 id，但是这里的图片要求是“纯 alpha 层”的图片，不然会变成一片灰色（参考：Android 通知栏微技巧，那些你所没关注过的小细节）
3. setAutoCancle 是用来设置点击通知之后通知自动消失
4. setColor 是用来设置小图标的颜色的
5. setContentIntent 是比较重要的，它设置了 PendingIntent，用来处理点击通知的事件。 这里的 PendingIntent 是一个包装 Intent 的类，它在被触发的时候才去启动内部的 intent。
6. 这里 PendingIntent 包裹的的 intent 用来开启详情页 activity，并传递过去商品对象（主界面随机发送过来的那个商品对象），第四个参数为 FLAG_UPDATE_CURRENT，表示 requestCode 一样的也能进行 intent 的更新

```
public class launchReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d( tag: "activity_create",  msg: "receive");
        GoodsItemBean itemBean = (GoodsItemBean) intent.getSerializableExtra(GoodsActivity.LAUNCH_BROADCAST_INFO);
        Intent intent1 = new Intent(context, DetailActivity.class);
        intent1.putExtra(GoodsActivity.JUMP_DETAIL_INFO, itemBean);
        PendingIntent pendingIntent = PendingIntent.getActivity(context,  requestCode: 0, intent1, PendingIntent.FLAG_UPDATE_CURR
        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
        builder.setContentTitle("新商品热卖")
                .setContentText(itemBean.name + "仅售" + itemBean.price)
                .setTicker("新消息")
                .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), itemBean.imageRes))
                .setSmallIcon(R.mipmap.fire_red)
                .setAutoCancel(true)
                .setContentIntent(pendingIntent)
                .setColor(Color.parseColor( colorString: "#f90101"));
        Notification notification = builder.build();
        NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        manager.notify( id: 0, notification);
    }
}
```

c. *manifest 文件中静态注册广播*

为上一步的 receiver 注册，过滤器能接受的 action 就是 mainactivity 中发的广播的 action

```
<receiver
    android:name=".receiver.launchReceiver">
    <intent-filter>
        <action android:name="com.lxc.my.LAUNCH"/>
    </intent-filter>
</receiver>
```

2. 点击购物车发布动态广播，产生“马上下单”的通知

a. *点击购物车发送广播，通过 eventbus 更新数据*

下图的函数在购物车点击的时候调用。 首先通过 eventbus 发送“加入购物车”事件，接着发布一个广播，这个广播接受的 action 为 BROADCAST_ACTION 这个常量对应的字符串，通过 intent 将当前详情页的商品对象传递到广播接收器，同时将“计数器 shopCnt”传入广播，待会有用处

```
/**
 * 物品加入购物车
 */
private void addToShopList(){

    AddToShopListEvent event = new AddToShopListEvent(goodsItem);
    EventBus.getDefault().post(event);

    Intent intent = new Intent(BROADCAST_ACTION);
    intent.putExtra(ADDITION_BROADCAST_INFO_ITEM, goodsItem);
    intent.putExtra(ADDITION_BROADCAST_INFO_INT, shopCnt);
    sendBroadcast(intent, receiverPermission: null);

    shopCnt++;

    Toast.makeText( context: this, text: "商品已添加到购物车", Toast.LENGTH_SHORT).show();

}
```

b. *重写 BroadcastReceiver，定义收到“加入购物车”广播的时候执行的动作*

setIcon，setContent 那些就不讲了，上面已经讲过。

这里同样用 pendingIntent 包裹一个 intent，用于跳转到商品页的 activity，intent 携带着一个 JUMP_GOODS_INFO 的信息，设置为 true 表明是广播跳转过去的

这里的 notify 函数的第一个参数（id）设置为“详情页活动”传进来的“计数器”值，这样就能使得每点击一次“购物车图标”就能产生一个通知

```
@Override
public void onReceive(Context context, Intent intent) {
    Log.d( tag: "activity_create",  msg: "receive");
    GoodsItemBean itemBean = (GoodsItemBean) intent.getSerializableExtra(DetailActivity.ADDITION_BROADCAST_INFO_ITEM);
    int id = intent.getIntExtra(DetailActivity.ADDITION_BROADCAST_INFO_INT,  defaultValue: 0);
    Intent intent1 = new Intent(context, GoodsActivity.class);
    intent1.putExtra(DetailActivity.JUMP_GOODS_INFO,  value: true);
    PendingIntent pendingIntent = PendingIntent.getActivity(context,  requestCode: 0, intent1,  flags: 0);
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
    builder.setContentTitle("马上下单")
            .setContentText(itemBean.name + "已加入购物车")
            .setTicker("新消息")
            .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), itemBean.imageRes))
            .setSmallIcon(R.mipmap.cart)
            .setAutoCancel(true)
            .setColor(Color.parseColor( colorString: "#303F9F"))
            .setContentIntent(pendingIntent);
    Notification notification = builder.build();
    NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
    manager.notify(id, notification);
```

由于不希望这里的跳转创建一个新的活动，因为这时候活动栈里面已经有 mainactivity 了。 所以这里将 launchMode 改为 singleTask，这样一来就能将栈顶的“详情页活动”出栈，从而 mainactivity 重新来到栈顶

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

        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

从通知跳转到由于使用 singleTask 模式因此 activity 重新回到栈顶的时候，会调用 onNewIntent()---->onResart()------>onStart()----->onResume()，~~因此这里在 onNewIntent()将新的 intent 设置进来，然后就可以在 onResart()里面使用这个 intent 了。--判断到这个 intent 是通知发过来的，就切换界面~~，用前面“划掉”的这种方法会有一个问题，就是直接从“详情页活动”finish 之后“商品页活动”重新显示出来也会调用 onRestart，于是直接 finish 而不是点击通知也会将界面切换到购物车界面。

解决方法是直接在 OnNewIntent 函数里面进行处理

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    boolean b = intent.getBooleanExtra(DetailActivity.JUMP_GOODS_INFO,  defaultValue: false);
    if (b){//如果是通知栏发过来的
        showShoppingCart( b: true);
    }
}
```

其中的 showShoppingCart 函数如下：

```
private void showShoppingCart(boolean b){
    if (b){
        fab_shop.setVisibility(View.GONE);
        fab_main.setVisibility(View.VISIBLE);
        recyclerView.setVisibility(View.GONE);
        fab_top.setVisibility(View.GONE);
        listView.setVisibility(View.VISIBLE);
    }
    else{
        fab_shop.setVisibility(View.VISIBLE);
        fab_main.setVisibility(View.GONE);
        recyclerView.setVisibility(View.VISIBLE);
        fab_top.setVisibility(View.VISIBLE);
        listView.setVisibility(View.GONE);
    }
}
```

c. *动态注册广播接收器以及 eventbus*

eventBus 注册与取消（在商品页的 activity 里面），接收到事件的时候添加一条商品信息

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(AddToShopListEvent event) {
    shopItems.add(event.getItemAdded());
    listAdapter.notifyDataSetChanged();
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getSupportActionBar().hide();
    setContentView(R.layout.activity_goods);

    EventBus.getDefault().register( subscriber: this);

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

“加入购物车的广播接收器”的注册与取消

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    getSupportActionBar().hide();
    super.onCreate(savedInstanceState);
    setContentView(R.layout.detail_layout);

    IntentFilter intentFilter = new IntentFilter(BROADCAST_ACTION);
    receiver = new additionReceiver();
    registerReceiver(receiver, intentFilter);

@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(receiver);
}
```

（3） 实验遇到困难以及解决思路

1. SingleTask 的 Activity 里面的 onRestart 函数调用 getIntent 得不到最新的 intent

解决：通过查看源码可知，虽然调用了 OnNewIntent 函数，但是父类 Activity 里面的 OnNewIntent 函数是空的（如下）

```
/**
 * This is called for activities that set launchMode to "singleTop" in
 * their package, or if a client used the {@link Intent#FLAG_ACTIVITY_SINGLE_
 * flag when calling {@link #startActivity}.  In either case, when the
 * activity is re-launched while at the top of the activity stack instead
 * of a new instance of the activity being started, onNewIntent() will be
 * called on the existing instance with the Intent that was used to
 * re-launch it.
 *
 * <p>An activity will always be paused before receiving a new intent, so
 * you can count on {@link #onResume} being called after this method.
 *
 * <p>Note that {@link #getIntent} still returns the original Intent.  You
 * can use {@link #setIntent} to update it to this new Intent.
 *
 * @param intent The new intent that was started for the activity.
 *
 * @see #getIntent
 * @see #setIntent
 * @see #onResume
 */
protected void onNewIntent(Intent intent) {
}
```

这里是空的

因此这时候在 OnRestart 里面 getIntent 得到的还是原本的 intent。
解决方法是：重写 onNewIntent 函数，手动将接收到的新的 intent 设置进去（setIntent），这样 intent 就更新了

四、 课后实验结果

设置通知的大图效果（BigPicture）

1. 效果展示

2. 代码

通过 setStyle 函数设置大图（setStyle 函数还可以设置长文本，点击才展开的文本等等）

```
Bitmap bm = BitmapFactory.decodeResource(context.getResources(), itemBean.imageRes);
builder.setContentTitle("新商品热卖")
        .setContentText(itemBean.name + "仅售" + itemBean.price)
        .setTicker("新消息")
        .setLargeIcon(bm)
        .setSmallIcon(R.mipmap.fire_red)
        .setAutoCancel(true)
        .setContentIntent(pendingIntent)
        .setStyle(new NotificationCompat.BigPictureStyle().bigPicture(bm))
        .setColor(Color.parseColor( colorString: "#f90101"));
```

五、 实验思考及感想

1. 开源库学习。这次文档介绍的 eventbus 就很好，使用发布/订阅模式，将解耦做得很彻底，github 上还有很多这样的开源库都能瞬间提高效率，比如 rxJava，okhttp，glide 之类几乎是安卓必备框架
2. 当遇到问题的时候，可以尝试看看源码进行分析，不仅提高阅读代码的能力，而且也是一种自我解决问题能力的培养，就像这次的 OnNewIntent 函数的问题，一看代码几乎就能明白了，况且源码一般都有很多详细的注释

作业要求：

1. 命名要求: 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。