移动应用开发实验报告

lab4:Broadcast使用

姓名	学号	年级	专业	电话	邮箱	起止日期
王若曦	15352319	15级	软件工程	15354222552	1511330303@qq.com	2017.10.26-2017.10.28

实验目的

1. 掌握Broadcast编程基础
2. 掌握动态注册Broadcast和静态注册Broadcast
3. 掌握Notification编程基础
4. 掌握EventBus编程基础

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。
具体要求：

1. 在启动应用时，产生一个通知，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，产生对应通知，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通知产生，点击购物车图标的通知由动态广播产生。

实验过程

整体思路分析

> 首先，通过静态广播产生一个随机商品的通知。这里要用到一个随机函数来生成一个1-10的随机数来选择商品，然后再MainActivity发送广播，同时静态广播要在AndroidManifest里进行注册。动态广播在商品详情界面点击购物车图标时发出，所以动态广播在商品详情界面注册。然后要给这两个广播定义一个接受类，并重写onReceive函数，在接收到通知后作出响应操作，即产生通知并实现点击通知进行的跳转等操作。

静态广播

首先是应用启动时的静态广播，这里首先通过一个随机函数生成一个1-10的整数用来选择商品。

```
int n=10;
Random random = new Random();
int m=random.nextInt( n );//把随机数random转化为int
```

然后把对应商品信息利用putputSerializable存到bundle里，然后利用putExtras把bundle存到intent里，再通过sendBroadcast把广播发送出去。

```
Intent intentBroadcast = new Intent(STATICACTION);//定义Intent,STATICACTION为静态广播接受
Bundle bundle = new Bundle();
bundle.putSerializable("goods",data.get(m));//data中第m个商品
intentBroadcast.putExtras(bundle);
sendBroadcast(intentBroadcast);
```

然后注册静态广播。静态广播的注册是在AndroidManifest文件里完成。

```
<receiver android:name="com.example.peter.lab3.MyStaticReceive">
    <intent-filter>
        <action android:name="MyStaticFilter"/>
    </intent-filter>
</receiver>
```

在application标签中，注册静态广播，其中第一个name是自定义的广播接受类，前面加上了路径，不过好像不加也可以；第二个name是自定义的静态广播action，这里的name的值就是我们要监听的静态广播的action，其实就是一个string，相当于一个key值。因为我们不可能任何广播都监听，所以需要一个key来过滤我们要监听的广播，即一个string字符串。这里的"MyStaticFilter"就是上面发送广播时给intent设的action：STATICACTION。

为了接受广播并弹出通知，我们要定义一个广播接受类MyStaticReceive（这里我一开始以为静态和动态的要分开写，后来就直接写在一起了，没有改名字）。然后重写onReceive方法，进行接受到广播后的操作。
这里首先定义静态广播和动态广播的action：

```
private static final String STATICACTION = "MyStaticFilter";//静态广播action
private static final String DYNAMICACTION = "MyDynamicFilter";//动态广播action
```

接着重写onReceive方法：因为我把静态和动态的写在了一起，所以先进行判断接受到的是静态广播还是动态广播；这里传进的参数是发送广播时的intent；然后利用NotificationManager和Builder来设置和弹出通知；这里builder有多种属性可供设置，如我们用到的setTitle设置标题,setText设置内容,setLargerIcon设置大图标等；弹出窗口之后我们还要实现点击通知的跳转，这里的通知是跳转到对应商品的详情界面，这里除了用到intent外，还用到了PendingIntent，一种特殊的intent，关于PendingIntent在后面详细介绍，这里只要知道我们是利用它和builder.setContentIntent进行点击通知从而执行Activity的跳转的;最后通过manager产生通知。

```
@Override
public void onReceive(Context context, Intent intent){
    if(intent.getAction().equals(STATICACTION)){
        Goods good = (Goods) intent.getExtras().get("goods");
        assert good != null;
        int image = good.getImageId();
        //大图标图片
        Bitmap bm = BitmapFactory.decodeResource(context.getResources(),image);
        //获取状态通知栏管理
        NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        //实例化通知栏构造器
        Notification.Builder builder = new Notification.Builder(context);
        //设置属性
        builder.setContentTitle("新商品热卖")//设置标题
                .setContentText(good.getName()+"仅售"+good.getPrice()+"!")
                .setTicker("您有一条新消息")
                .setLargeIcon(bm)
                .setSmallIcon(image)
                .setAutoCancel(true);
        //绑定intent，点击图标跳转到商品详情页
        Intent mIntent = new Intent(context,Goods_info.class);
        mIntent.putExtras(intent.getExtras());
        PendingIntent mPendingIntent = PendingIntent.getActivity(context,0,mIntent,PendingIntent.FLAG_UPDATE_CURRENT);
        builder.setContentIntent(mPendingIntent);
        //绑定Notification,发送通知请求
        Notification notify = builder.build();
        manager.notify(0,notify);
    }
    ...//动态
}
```

动态广播

动态广播是在商品详情界面点击购物车图标时发出的，所以我们在商品详情页Goods_info里进行广播的发出和注册。
首先，定义动态广播的action：

```
private static final String DYNAMICACTION = "MyDynamicFilter";//动态广播action
```

然后利用此界面中的goods作为传递信息封装到intent里，然后sendBroadcast发送出去。

```
Intent intentBroadcast = new Intent(DYNAMICACTION);//定义Intent
Bundle bundle = new Bundle();
bundle.putSerializable("goods",goods);
intentBroadcast.putExtras(bundle);
sendBroadcast(intentBroadcast);
```

注册动态广播接收者。动态广播的注册是在代码中完成的，所以我们要在这里进行动态广播接收的注册。这里主要是把我们定义的动态广播的action作为key添加到过滤器中，用来过滤我们想要监听的广播。

```
void registerRec(){
    mReceiver = new MyStaticReceive();
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(DYNAMICACTION);
    registerReceiver(mReceiver,intentFilter);
}
```

这里我还手动注销了该动态广播：

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

}
```

然后是为动态广播重写onReceive方法。动态的方法和静态的差不多，区别在于通知的内容，跳转的界面以及弹出通知的次数和不同。

```
int id;
int time=0;
@Override
public void onReceive(Context context, Intent intent){
    ...//静态
    else if(intent.getAction().equals(DYNAMICACTION)){

        Goods good = (Goods) intent.getExtras().get("goods");
        id = good.getId();
        assert good != null;
        int image = good.getImageId();
        Bitmap bm = BitmapFactory.decodeResource(context.getResources(),image);
        //获取状态通知栏管理
        NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        //实例化通知栏构造器
        Notification.Builder builder = new Notification.Builder(context);
        //设置属性
        builder.setContentTitle("马上下单")//设置标题
                .setContentText(good.getName()+"已添加到购物车")
                .setTicker("您有一条新消息")
                .setLargeIcon(bm)
                .setSmallIcon(image)
                .setAutoCancel(true);
        //绑定intent，点击图标跳转到商品详情页
        Intent mIntent = new Intent(context,MainActivity.class);
        Bundle bundle = new Bundle();
        bundle.putSerializable("goods",good);
```

```
            mIntent.putExtras(bundle);
            PendingIntent mPendingIntent = PendingIntent.getActivity(context,id,mIntent,0);
            builder.setContentIntent(mPendingIntent);
            //绑定Notification,发送通知请求
            Notification notify = builder.build();
            manager.notify(id+time,notify);
            time++;
        }

    }
```

这里的优化是，多次点击同一件商品的购物车图标会弹出多次通知，点击不同商品的购物车图标弹出的通知不会覆盖之前弹出的通知：
因为manager.notify(id+time,notify)这个方法产生通知，其中第一个参数是弹出通知的id，也就是说每次弹出的通知如果id是相同的话，那么新的通知就会覆盖旧的通知，导致旧的通知消失。所以我一开始是在这里定义了一个time变量，用来记录该商品点击了几次，每点一次便加一，然后以它为id进行通知产生。但是这样对每个商品都是一样的，都会从0计数，也就是说第一个商品添加到购物车的通知会被第二个商品添加到购物车的通知覆盖。所以最后我又在Goods里面给每个商品定义了一个id，用来区分不同的商品，10个商品的id从100-1000，这样产生通知的时候以id+time为通知的id，这样在100次通知内是不会覆盖的（怎么会有人手机里有100条相同通知还能忍的）。
还有是点击这里的通知会直接跳转到购物车界面，也就是跳到MainActivity之后要设置ListView 可见，RecyclerView 不可见，所以我们要在MainActivity声明onNew Intent方法，进行该操作。即从其他Activity跳回MainActivity时会显示购物车列表而不是商品列表。

```
@Override
protected void onNewIntent(Intent intent){
    mRecyclerView.setVisibility(View.GONE);
    mListView.setVisibility(View.VISIBLE);
    fab.setImageResource(R.drawable.mainpage);
}
```

EventBus实现数据更新

这次实验利用EventBus实现购物车的添加和更新。EventBus简化了应用程序内各组件间、组件与后台线程间的通信。优点是开销小，代码更优雅，以及将发送者和接收者解耦。
所以，像前面的RecyclerView，EventBus也要添加依赖，不过这次添加依赖很顺利，没有乱七八糟的报错。
这里我并没有给EventBus新建一个事件类，而是直接用Goods来进行数据的传递，因为没有必要新写一个，直接用Goods就行。
EventBus的数据传递是在商品详情界面点击购物车图标时发生，即代替了原来的setResult。但是没有结果码，所以处理的事件我感觉比起setResult和onActivityResult单一一些。
EventBus的数据传递很简单，`EventBus.getDefault().post(goods);`直接把对应商品goods传递出去。
然后再MainActivity即购物车所在Activity接收传递过来的信息，同时要在该Activity注册订阅
者`EventBus.getDefault().register(this);`，并写订阅方法，即接受信息后的响应操作：

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void omMessageEvent(Goods c) {
    Map<String,Object> listItem = new LinkedHashMap<>();
    assert c != null;
    listItem.put("first",c.getFirstLetter());
    listItem.put("name",c.getName());
    listItem.put("price",c.getPrice());
    shoplist.add(c);
    shoppingitem.add(listItem);
    simpleAdapter.notifyDataSetChanged();
}
```

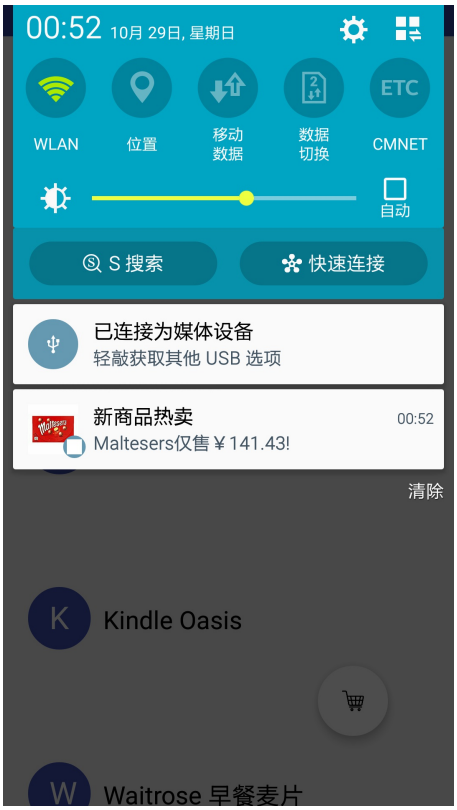
这里的注解@Subscribe和后面的线程模型MAIN是指事件的处理会在UI线程中执行。事件处理时间不能太长，长了会ANR的。具体还不是很清楚，只是声明订阅方法时要这么写。
里面具体的操作和原来的onActivityResult基本一致，只是我们不需要在从intent里取出goods，因为这里EventBus直接传的就是

goods。
和注销动态广播一样，这里也要注销EventBus，注销方法同样在onDestroy里实现。

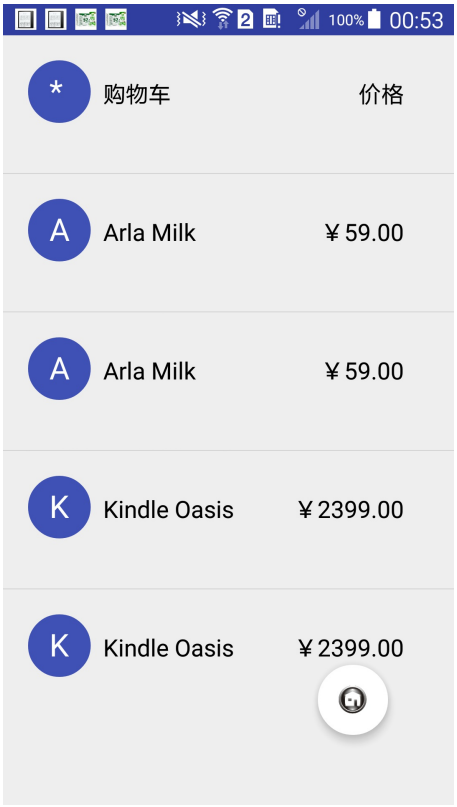
```
@Override
protected void onDestroy(){
    super.onDestroy();
    EventBus.getDefault().unregister(this);
}
```

实验结果

静态广播：



动态广播：



实验总结

实验中遇到的问题：

1. 静态广播的注册。静态广播的注册是在AndroidManifest文件里完成，但是一开始不知道要写在哪个标签里，所以直接写在了MainActivity标签里。然后接收不到广播，后来查了之后发现是要写在Activity标签外面，application标签里面。还有一点比较坑爹的就是AndroidManifest文件里是不会报错的，所以写错了也不知道哪里有错误。

2. 通知的点击跳转。实现点击通知进行跳转时，要用到PendingIntent和setContentIntent方法。pendingIntent是一种特殊的Intent。主要的区别在于Intent的执行立刻的，而pendingIntent的执行不是立刻的。pendingIntent执行的操作实质上是参数传进来的Intent的操作，但是使用pendingIntent的目的在于它所包含的Intent的操作的执行是需要满足某些条件的。
   主要的使用的地方和例子：通知Notificatio的发送，短消息SmsManager的发送和警报器AlarmManager的执行等等。它们两者的结合可以为Notification处理多种事件，如这里的跳转事件。具体更多用法就不在这里详述，贴一个博客供参考：`http://blog.csdn.net/yczz/article/details/28416893`
3. EventBus的事件类。这里我没有单独为EventBus重新定义一个事件类，因为用Goods做它的事件类就很好，直接传goods，也省掉了从intent等中取出goods的步骤。因为传递的实质就是对应goods。
4. onNew Intent的使用。一开始点击动态广播的通知时不知道要怎么直接跳转到购物车列表，因为MainActivity的主界面设置的是商品列表。后来查到了onNew Intent方法， 设置跳转到该Activity时的相应操作，这里直接把商品列表设为不可见，购物车列表可见即可。
5. 同一商品点击多次购物车图标弹出多个通知，不同商品点击多次购物车图标弹出不同通知。这里要通过设manager调用notify时的id来设置弹出不同通知。为了对同一商品多次点击弹出多个通知，可以通过添加一个time变量每次++来实现；为了实现不同商品弹出通知不被覆盖，我给Goods类中加了一个id属性，即给每个商品添加了一个id，用来区分不同商品，这里id的区间为100。所以最后利用id+time作为notify的id来进行通知的产生。这样会在100条通知范围内不会被覆盖。
6. Activity的launchmode。这次真的是很坑，实验文档上说要把launchmode改为singleInstance，来保证点击通知后不会新建一个购物车列表，但是这样还是会产生新的购物车列表。具体的bug是，启动后点击随机的通知，跳转后点击购物车图标，然后再点击通知，然后切换到商品列表，点开任意一个商品点击购物车图标，这时会弹出两个通知。
   所以这里应该把launchmode改为singleTask。关于这两个launchmode：

> “singleTask”和“singleInstance”的activity只能启动一个task。它们通常在activity栈的根（root）上，此外（moreover），设备中同一时间只保持唯一的一个activity实例---只有一个这样的栈。
> “singleTask”和“singleInstance”模式只在一个方面有区别：“singleTask” activity允许其他activity成为task中的一部分，并且它通常在task栈的根部，其它activity（必须是“standard”和“singleTop”模式的activity）可以加载到这个栈上。另一方面，“singleInstance”模式的activity不允许任何其它activity加载到这个task栈上。它是这个task中唯一的activity。如果它启动另一个activity，那个activity会被指引到不同的task中，这个行为就像在intent中用FLAG_ACTIVITY_NEW_TASK标记一样。

然而对这两个launchmode还是不是很理解，下次实验课上准备问一下TA。

这次实验相对于实验三来说简单了很多，实现静态通知和动态通知即可，然后利用EventBus代替setResult和onActivityResult实现购物车的添加。有一点值得注意的是关于AndroidManifest文件的修改，比如静态广播的注册是在这里，launchmode的选择也是在这里，还有启动图标即app的图标等等，这里好像能做的事情有很多。最后想说，TA大大们如果更新了实验文档可不可以在群里发个公告，这次让launchmode坑了一个小时，有点伤。