# 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 | 15M1 | 专业（方向） | 移动互联网 |
| 学号 | 15352008 | 姓名 | 蔡荣裕 |
| 电话 | 13727021990 | Email | 897389207@qq.com |
| 开始日期 | 2017/10/24 | 完成日期 | 2017/10/30 |

## 实验题目

appwidget 及broadcast 使用

## 实验目的

1. 掌握 AppWidget 编程基础
2. 掌握 Broadcast 编程基础
3. 掌握动态注册 Broadcast 和静态注册 Broadcast

## 实验内容

实现一个Android 应用，实现静态广播、动态广播两种改变widget 内容的方法。

|  |  |  |  |
| --- | --- | --- | --- |
| 1. widget 初始情况如右图1。 2. 点击widget可以启动应用，并在widget随机推荐一个商品，如图2 3. 点击widget跳转到该商品详情界面，如图3 |  |  |  |
| 图1 | 图2 | 图3 |
| 1. 点击购物车图标，widget相应更新如图4 2. 点击widget跳转到购物车界面，如图5 3. 实现方式要求:启动时的widget的更新通过静态广播实现，点击购物车图标时候widget的更新通过动态广播实现。 | |  |  |
|  | | 图4 | 图5 |

## 课堂实验结果

1. 实验截图：
   * 1. 虚拟机上效果如下：

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. 关键步骤：
2. 在启动应用时，会有通知产生，随机推荐一个商品
   1. 首先是需要产生一个随机数，然后将这个随机数作为参数发出去，因为已经前一个实验已经实现数据类。

|  |
| --- |
| private static final String *STATICACTION* = "cn.chonor.lab4.staticreceiver";  private void broad\_init(){  Random random=new Random();  int i=random.nextInt(10);  Intent iBroad=new Intent(*STATICACTION*);  Bundle bundle = new Bundle();  bundle.putInt("position", i);  iBroad.putExtra("mainActivity",bundle);  sendBroadcast(iBroad); } |

STATICATION为自己设定的广播名称。由于是静态注册所以需要在Mainfest.xml 中进行注册，同时为了防止重复生成Main这个页面还需要修改launchMode（其他两个界面同时修改）。

|  |
| --- |
| <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>  <receiver android:name=".MainActivity$StaticReceiver" android:enabled="true">  <intent-filter>  <action android:name="cn.chonor.lab4.staticreceiver" />  </intent-filter> </receiver> |

此处静态广播就注册了，因为我们把广播写到MainActivity所以此处需要用.MainActivity$StaticReceiver

* 1. 在静态广播类StaticReceiver 中重写onReceive方法，当接收到对应广播时进行数据处理，产生通知。

此处要用Notification.Builder，动态设置Notificatio。

|  |
| --- |
| public static class StaticReceiver extends BroadcastReceiver {  public StaticReceiver(){}  @Override //静态广播接收器执行的方法  public void onReceive(Context context, Intent intent) {  if(intent.getAction().equals(*STATICACTION*)) {  Bundle extras = intent.getBundleExtra("mainActivity");  Data data=new Data();  int id=extras.getInt("position"); //获得数据位置  Bitmap bm=BitmapFactory.*decodeResource*(context.getResources(),data.ID[id]);  String name=data.getGood\_list\_index(id).getGoodName();  String price=data.getGood\_list\_index(id).getGoodPrice();   NotificationManager notifyManager = (NotificationManager) context.getSystemService(Context.*NOTIFICATION\_SERVICE*);   Intent intent1 =new Intent (context,Good\_Info.class); //点击事件和传输  Bundle bundle = new Bundle();  bundle.putInt("position", id);  bundle.putParcelableArrayList("data",data.getGood\_list());  bundle.putParcelableArrayList("cart", data.getCart\_list());  intent1.putExtra("mainActivity", bundle);  PendingIntent pi = PendingIntent.*getActivities*(context, 0, new Intent[]{intent1}, PendingIntent.*FLAG\_CANCEL\_CURRENT*);   //实例化NotificationCompat.Builde并设置相关属性  Notification.Builder builder = new Notification.Builder(context)  //设置小图标  .setSmallIcon(data.ID[id])  .setLargeIcon(bm)  //设置通知标题  .setContentTitle("新商品热卖")  //设置通知内容  .setContentText(name+"仅售"+price+"!")  .setContentIntent(pi)  .setAutoCancel(true);  notifyManager.notify(0, builder.build());  }  } } |

这样我们启动后的推荐就完成了。

1. 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。
   1. 实现 BroadcastReceiver子类，并且重写onReceive方法，与静态广播类中类似，此处因为需要先打开购物车界面注册，但是一开始没有打开，所以此处广播还是使用Intent传参（课后实验中使用粘性Eventbus解决，不需要intent传参）。

|  |
| --- |
| public class DynamicReceiver extends BroadcastReceiver {  public static final String *DYNAMICATION*= "cn.chonor.lab4.dynamicreceiver";  public DynamicReceiver(){}  @Override  public void onReceive(Context context, Intent intent) {  if(intent.getAction().equals(*DYNAMICATION*)){  NotificationManager notifyManager = (NotificationManager) context.getSystemService(Context.*NOTIFICATION\_SERVICE*);   Intent intent1 =new Intent (context,Shoppingcart.class);//点击事件和传输  Bundle bundle = new Bundle();  bundle.putParcelableArrayList("data", data.getGood\_list());  bundle.putParcelableArrayList("cart", data.getCart\_list());  intent1.putExtra("mainActivity", bundle);  PendingIntent pi = PendingIntent.*getActivities*(context, 0, new Intent[]{intent1}, PendingIntent.*FLAG\_CANCEL\_CURRENT*);  //实例化NotificationCompat.Builde并设置相关属性  Notification.Builder builder = new Notification.Builder(context)  //设置小图标  .setSmallIcon(data.ID[id])  .setLargeIcon(BitmapFactory.*decodeResource*(context.getResources(),data.ID[data.map.get(names)]))  //设置通知标题  .setContentTitle("马上下单")  //设置通知内容  .setContentText(names+"已添加到购物车")  .setContentIntent(pi)  .setAutoCancel(true);  notifyManager.notify(0, builder.build());  }  } } |

进行动态广播注册，我们只需要在onCreate的时候调用一下

|  |
| --- |
| public static final String *DYNAMICATION*= "cn.chonor.lab4.dynamicreceiver";  private DynamicReceiver dynamicReceiver=new DynamicReceiver(); private IntentFilter dynamic\_filter =new IntentFilter();  private void broad\_init(){  dynamic\_filter.addAction(*DYNAMICATION*);  registerReceiver(dynamicReceiver,dynamic\_filter); } |

之后我们在onDestroy()进行销毁

|  |
| --- |
| @Override  protected void onDestroy() {  super.onDestroy();  unregisterReceiver(dynamicReceiver); } |

* 1. 然后我们只要在，购物车点击事件中增加发送广播，此处还有EventBus的传输。

|  |
| --- |
| cart.setOnClickListener(new View.OnClickListener() { //购物车按钮  @Override  public void onClick(View view) {  if(flag){  int incart= -1; //确定购物车里有没有  for(int i=0;i<data.getCart\_list().size();i++){  if(data.getGood\_list\_index(id).getGoodName().equals(data.getCart\_list\_index(i).getGoodName()))  incart=i;  }  if(incart== -1) //没有加入购物车  data.addCart\_list(data.getGood\_list\_index(id));  else //有数量+1  data.setCart\_list\_Cnt(incart,data.getCart\_list\_index(incart).getCnt()+1);  Toast.*makeText*(Good\_Info.this,"成功将"+data.getGood\_list\_index(id).getGoodName()+"加入购物车",Toast.*LENGTH\_SHORT*).show();  }  else {  data.setCart\_list\_Cnt(id,data.getCart\_list\_index(id).getCnt()+1);  Toast.*makeText*(Good\_Info.this,"成功将"+data.getCart\_list\_index(id).getGoodName()+"加入购物车",Toast.*LENGTH\_SHORT*).show();  }  EventBus.*getDefault*().post(new MessageEvent(data)); //发送数据  Intent iBroad=new Intent(*DYNAMICATION*);//发送广播  sendBroadcast(iBroad);  } }); |

* 1. 因为此处我们要用Eventbus进行数据传输所以我们需要在购物车界面中注册Eventbus，首先在onCreate中注册

|  |
| --- |
| public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*shopping*);  init();//初始化  get\_set\_data();//接收数据  init\_listener();//事件监听  EventBus.*getDefault*().register(this); } |

之后onDestroy()注册取消。

|  |
| --- |
| @Override  protected void onDestroy() {  super.onDestroy();  //取消注册事件  EventBus.*getDefault*().unregister(this); } |

* 1. 之后在购物车中写好订阅

|  |
| --- |
| @Subscribe(threadMode = ThreadMode.*MAIN*)  public void onMessageEvent(MessageEvent event){  data=event.getData(); } |

* 1. 此处我们只需要简单的实现一下事件类，用来传递我们的Data类

|  |
| --- |
| public class MessageEvent {  private Data data;  public MessageEvent(Data data) {  this.data = data;  }  public Data getData() {  return data;  }  public void setData(Data data) {  this.data = data;  } } |

至此我们的动态广播和数据传输就设置好了。

1. 实验中遇到的困难和解决思路：
2. 一开始无法接受到静态和动态广播，之后发现这个是8.0的权限要求问题，换回7.1.1即可使用。
3. 在使用launchMode为singleTask时候，发现不会更新数据，然后百度了一下因为此时不会再次执行onCreate函数，但是onNewIntent可以解决问题。

1. 使用Eventbus时候发现这个页面需要先打开注册一次，因为之前实验中使用了3页面，所以放弃了全部使用Eventbus传参，还是留了一部分Parcelable接口。但是在课后实验中发现可以直接使用Sticky Eventbus进行实现，这部分在课后拓展。
2. 对于实验中给出的思考题大ICON 如何设置？bm 是什么？ bm实际是bitmap，只需要转化一下就好了。

## 课后实验结果

* + 1. 使用Sticky Eventbus进行传参，解决了3页面时购物车页面没有打开造成Eventbus没有注册进而收不到消息。相对于之前的代码可以实现购物车按钮和界面的完全同步，只需要修改两个地方，然后我们就可删除之前的Intent传递数据部分了

首先是Eventbus的发送

|  |
| --- |
| EventBus.*getDefault*().postSticky(new MessageEvent(data)); //发送数据 |

之后是Eventbus的接收

|  |
| --- |
| @Subscribe(threadMode = ThreadMode.*POSTING*,sticky = true)  public void onMessageStickyEvent(MessageEvent event){  data=event.getData();  get\_set\_data();//重设数据 } |

* + 1. 优化了一下弹窗模式

如下图，点击这个弹出的通知栏可以直接转跳，并增加了详细信息。

|  |  |  |
| --- | --- | --- |
|  |  |  |

代码修改如下：

|  |
| --- |
| Notification.InboxStyle inboxStyle = new Notification.InboxStyle()  .setBigContentTitle("新商品热卖")  .addLine(name+"仅售"+price+"!")  .addLine(data.getGood\_list\_index(id).getGoodTypes() + " " + data.getGood\_list\_index(id).getGoodInfo());  Notification.Builder builder = new Notification.Builder(context)  //设置小图标  .setSmallIcon(data.ID[id])  .setLargeIcon(bm)  //设置通知标题  .setTicker("TickerText:" + "您有新短消息，请注意查收！")  .setContentTitle("新商品热卖")  //设置通知内容  .setContentText(name+"仅售"+price+"!")  .setStyle(inboxStyle)  .setContentIntent(pi)  .setAutoCancel(true)  .setDefaults(Notification.*DEFAULT\_SOUND* | Notification.*DEFAULT\_VIBRATE*)  .setWhen(System.*currentTimeMillis*())  .setPriority(Notification.*PRIORITY\_HIGH*)  .setVisibility(Notification.*VISIBILITY\_SECRET*); |

## 实验思考及感想

这次实验比较简单只需要在上一个实验的基础上进行修改，不需要太多的处理，广播部分比较简单，就是注册和发送，但是一直搞不懂为什么8.0的系统怎么样都接收不到，但是换到7.1.1之后正常，在8.0的系统中额外增加了获取权限也是没有成功实现。

之后就是Eventbus的使用，因为上一个实验中使用了3页面的模式，所以造成了普通的Eventbus，无法再购物车界面未启动的时候传输数据，但是最后使用了粘性Eventbus处理了这个问题，使得数据能够正常传输。

最后的部分给整个广播系统增加了一下弹出功能和声音震动，优化一下使用体验。