中山大学移动信息工程学院本科生实验报告

年级	1501	专业（方向）	移动（互联网）
学号	15352015	姓名	曹广杰
电话	13727022190	Email	1553118845@qq.com

Content

一、实验题目

Appwidget及Broadcast 使用

二、实现内容

本次实验模拟实现一个商品列表。

- 掌握 AppWidget 编程基础
- 掌握 Broadcast 编程基础
- 掌握动态注册 Broadcast 和静态注册 Broadcast

实现方式要求:

- 点击widget可以启动应用，并在widget随机推荐一个商品
- 点击widget跳转到该商品详情界面
- 点击购物车图标，widget相应更新
- 点击widget跳转到购物车界面
- 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

三、课堂实验结果

实验截图

对于widget的处理，从触发步骤来讲，和动态/静态广播是一样的，使用的同样的触发方式——都是需要有一方发送信息，在接收端接收信息并作出相应的反应。

实验步骤以及关键代码

广播有几个特性：发送面向全体成员，不负责接收器行为，接收端针对指定的广播信息执行指定的行为。

为此，需要实现：

- 广播发生器
- 广播接收器
- 广播接收器的过滤机制注册

为widget建立静态广播

静态广播的管理比较智能化和简单化，静态广播属于常驻型广播，在关闭该应用之后依然存在，这种情况下，很显然需要系统的权限与来自系统的服务——因此需要对静态广播进行注册。

注册静态广播

在新建widget之后，系统会自动在AndroidManifest.xml中用`<;receiver>;`标签进行注册，并在标签内用`<;intent- filter>;`标签设置过滤器。

```
<receiver android:name=".widget">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
</receiver>
```

- 这里第二个参数action是需要监听的消息的名字。

  接收器会对系统空间中存在的各种各样的信息进行过滤，只选择名为“安卓.app挂件.行为.APP挂件更新”的广播进行接收处理。名字是有点长，不过既然是系统初始化的，复制粘贴吧。

- 这里第一个name是静态广播接收器的名字。

  静态广播接收器也需要一个标志的，静态广播的名字就起到这么一个作用，静态广播的名字就是我们写的widget的类名字。

静态广播传递数据

静态广播用于发送广播，而发送广播也是需要有内容的，对于静态广播我们使用intent储存静态广播所包含的一系列数据。

```
    // 静态广播的名字
String WIDGET_STATIC = "android.appwidget.action.APPWIDGET_UPDATE";
// 把名字放在intent中
Intent widget_intent = new Intent(WIDGET_STATIC);
widget_intent.putExtras(bundle);// 插入一些widget需要的数据
sendBroadcast(widget_intent);// 发送
```

这样，静态广播发生器就发送了这一次的广播内容。由于sendBroadcast函数是在MainActivity中调用的，这就说明，一旦该APP打开，则广播发送。

接收静态广播

静态广播发送出之后，我们在系统中注册了静态广播的接收器。

- 静态广播发送的信息标志是：“android.appwidget.action.APPWIDGET_UPDATE”
- 而我们定义的静态广播接收器的信息标志也是：“android.appwidget.action.APPWIDGET_UPDATE”
- 那么，接收器在感知到广播的消息后就启动了

接收到静态广播之后，应该采取一些行动的。但是采取什么行动，在广播中是没有定义的，这正是我们之前所提及的，广播仅仅负责发送信息，接收方的行为不在考虑范围之内，也正是因此，在实现基类的时候，设置onReceive()函数为@Override，意为一定要重新实现一下。

重新实现onReceive()函数

我们希望在接收方（这里因为是在系统中注册的，所以是系统）接收到广播的信息之后，可以采取相应的行为，那么就需要在接收之后的部分添加内容：

```
public class widget extends AppWidgetProvider {
    private static final String WIDGET_STATIC
      = "android.appwidget.action.APPWIDGET_UPDATE";

    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals(WIDGET_STATIC)) {具体行为}
```

这里的具体行为就是更新widget的信息啦。由于widget并不在自己的界面上，而是一个居住于宿主程序（这里是系统程序）的小挂件，那么很显然，我们是不能直接操作系统的程序的，为此，我们需要一些信使去实现传递信息的功能。

先准备一个包，然后把我们要传输的信息打包。由于我们要传输到宿主的系统，就需要一个能长途跋涉的包——在修改系统时可以调用的包：

```
// 准备包
RemoteViews static_widget =
     new RemoteViews(context.getPackageName(), R.layout.widget);
// 打包信息
static_widget.setTextViewText(R.id.widget_text, name + " only " + price);
static_widget.setImageViewResource(R.id.widget_image, imageId);
static_widget.setOnClickPendingIntent(R.id.widget, widget_chge);
```

接下来，就需要填写发件人的书名以及地址了，这里就需要一个类componentName：

```
ComponentName componentName =
        new ComponentName(context, widget.class);
```

即便是如此，我们依然不能与widget相见，需要经过其管理者，将信息转达给widget：

```
AppWidgetManager appWidgetManager
        = AppWidgetManager.getInstance(context);
appWidgetManager.updateAppWidget(componentName, static_widget);//（发件人， 信息包）
```

通知栏的界面切换

如果希望在点击通知界面的时候，界面可以跳转到一个新的界面。需要使用一个类：`PendingIntent`：

```
PendingIntent widget_chge =
    PendingIntent.getActivity(context, 0, 富含信息的intent,
                              PendingIntent.FLAG_UPDATE_CURRENT);
static_widget.setOnClickPendingIntent(R.id.widget, widget_chge);
// R.id.widget 表当前widget界面的布局
```

Pendingintent表示尚未成熟的intent，只需要按照参数要求传入参数，就可以运行。这个部分放在了设置通知builder的前面，而在通知出现之后，该intent也没有马上作用，而是等到我们点击通知的时候，Pendingintent成熟，继而实现跳转。

从widget跳转需要调用函数setOnClickPendingIntent()，实现从当前界面跳转到之后界面的功能。

建立动态广播

动态广播的注册方式不同于静态广播，动态广播不需要在Manifest.xml文件里声明，而是在MainActivity函数中声明，与静态广播一样，也需要定义广播接收器以及过滤信息：

```
// 接收器的接受信息限制
IntentFilter dynamic_widget =
      new IntentFilter("android.appwidget.action.dynamic.APPWIDGET_UPDATE");
// 注册
registerReceiver(new widget(), dynamic_widget);
```

发送与接收动态广播

之前是在MainActivity中发送广播的，而这一次如果我们需要在特定的情况下发送，就可以在其他界面的某一个函数下调用sendBroadcast函数。

接收广播时候也一定会采取特殊的行为，该过程与静态设置一致。因为接收器都是同一个类，都是调用同一个onReceive函数，故而可以应对静态的和动态的广播接收，即在onReceive()函数中添加一个条件语句即可：

```
public class widget extends AppWidgetProvider {
    private static final String WIDGET_STATIC
      = "android.appwidget.action.APPWIDGET_UPDATE";
    private static final String WIDGET_DYNAMIC
      = "android.appwidget.action.dynamic.APPWIDGET_UPDATE";

    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals(WIDGET_STATIC)) {具体行为}
        if(intent.getAction().equals(WIDGET_DYNAMIC)) {具体行为}
```

实验遇到困难以及解决思路

动态广播注册的方法

```
IntentFilter dynamic_widget =
        new IntentFilter("android.appwidget.action.dynamic.APPWIDGET_UPDATE");
registerReceiver(new widget(), dynamic_widget);
```

之前注册的时候，这里总是出现问题，接收端总是接收不到广播。后来发现原来是注册的时候出的问题，由于注册的代码时候直接由动态广播复制粘贴的，在register的第一个参数中是动态广播的实例化——而此处应该是widget的实例化，因为我们需要注册widget以及其接收器过滤器。

onNewIntent不能实时更新的问题

如果发一个通知给状态栏，然后点击这个通知，自然会执行 PendingIntent 里边的Intent。

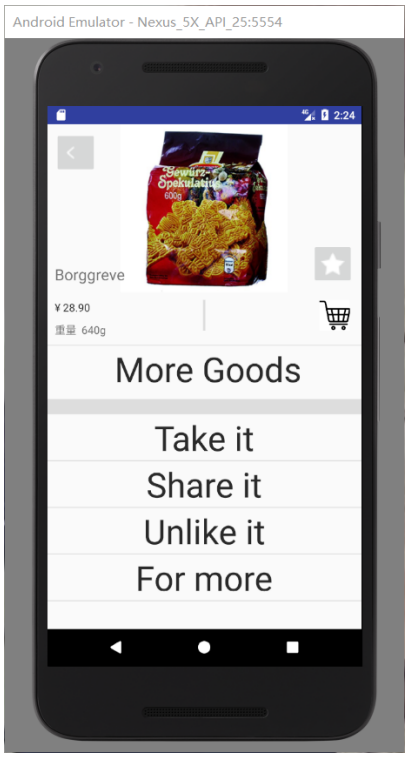
但是，有一次在Activity那边的 onNewIntent()方法里边得到的数据不是最新的——在使用PendingIntent的时候（就是延迟启用的intent），需要在PendingIntent的后面添加一个Flag：

```
PendingIntent Jump_java
        = PendingIntent.getActivity
            (context, 0, To_java, PendingIntent.FLAG_UPDATE_CURRENT);
```

四、课后实验结果

标记星信息的同步化

本次实验笔者依然延续了之前对于标记星的处理，就是无论在哪一个界面点击了标记星，在进入另一个界面的时候，都可以实现同步：



五、实验思考及感想

本次实验的主要任务是完成静态广播以及动态广播对widget的设计。

如果需要实现界面的转移：

- Intent关联用于切换的界面
- PendingIntent用于，继续intent的实现，显示切换的内容
- 切换的界面如果已经存在则使用onNewIntent函数配合单任务注册，避免重建
- 调用PendingIntent的时候，在widget的实现中是使用setOnClickPendingIntent实现跳转

widget的布局问题，内容问题，触发问题都是相对独立的，完全可以各自独立地完成。此外，设计还是相对独立的，至于内容的设定则是在重构函数内部实现。这就是开发体系结构上的层次化，使得开发过程结构清晰，分级完整。而拼接组合也非常有趣，尽管以上面对的函数功能千差万别，或者是系统本身的，或者是自己定义的，但是它们在传入数据的时候，都在遵循函数运行的基本架构——如函数sendBroadcast的使用，就是简单清晰、容易理解的。至于Intent_filter的过滤，有很大一部分并不需要我们去操作。