移动应用开发实验报告

lab5:appwidget 及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	2017.10.30-2017.10.31

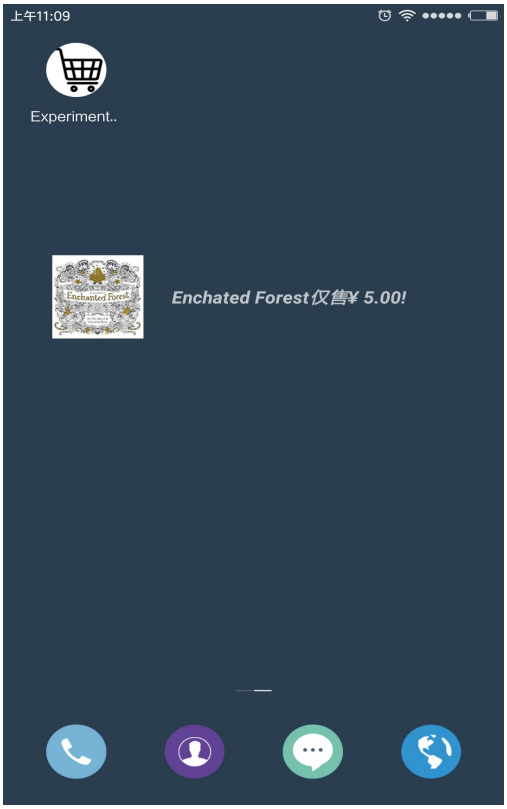
实验目的

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

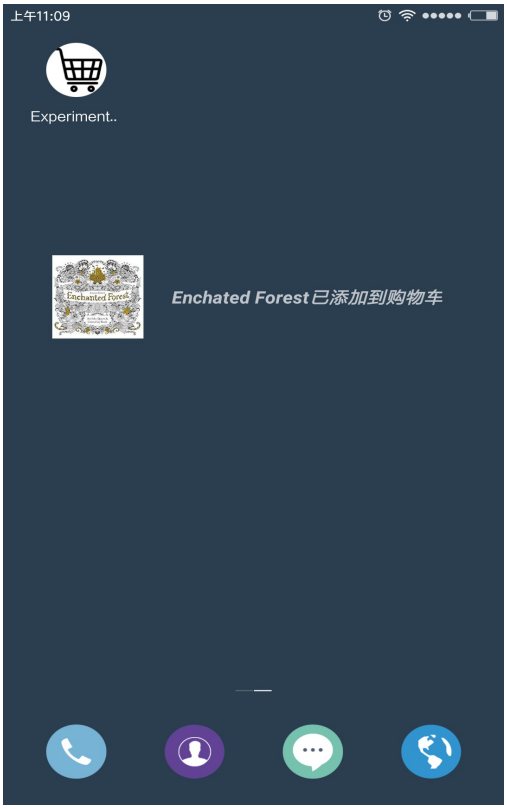
实现一个Android 应用，实现静态广播、动态广播两种改变w idget 内容的方法。在上次实验的基础上进行修改，所以一些关于静态动态广播的内容会简略。
具体要求:
(1)w idget 初始情况如下：

(2)点击w idget可以启动应用，并在w idget随机推荐一个商品:

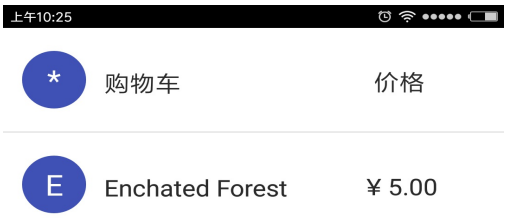


(3)点击w idget跳转到该商品详情界面:

(4))点击购物车图标，w idget相应更新:



(5)点击w idget跳转到购物车界面:



(6)实现方式要求:启动时的w idget的更新通过静态广播实现，点击购物车图标时候w idget的更新通过动态广播实现。

实验过程

整体思路分析：

> 本次实验添加一个屏幕小组件w idget,用于接受静态和动态广播，并做出对应响应。由于Widget是Android Studio自带的一个类，创建对应Widget类后，系统会自动生成相关文件：java，xml布局文件和xml内容提供者文件等。所以我们只需要补全我们需要的代码实现响应功能即可。首先，修改布局文件实现我们想要的布局效果；然后修改内容提供者xml文件进行Widget内容初始化；然后在java文件中修改相关的onupdate,并重写onReceive等方法，进行广播的接受和数据的更新以及点击事件的处理。

xml布局文件

布局文件利用一个ImageView 和一个TextView 组成，这里自动生成的xml文件是Relative布局的。这里要注意在布局中加入id,用于后面处理点击事件时使用，即点击该小组件进行相应处理，而不是点击其中的图片或文字。这里有一点是设置TextView 背景透明，通过设置背景色为#00000000实现。然后将两个控件垂直对中，利用margin设置位置。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/mWidget"
    >
    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:id="@+id/pic"
        android:src="@drawable/shoplist"
        android:layout_marginLeft="0dp"
        android:layout_centerVertical="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:id="@+id/appwidget_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:background="#00000000"
        android:contentDescription="@string/appwidget_text"
        android:textColor="#ffffff"
        android:textSize="15sp"
        android:textStyle="bold|italic"
        android:layout_marginLeft="120dp"

        />
</RelativeLayout>
```

关于布局这里要说一下，Widget在手机中可以手动调整宽度，通过拖拉实现。所以跑出来之后布局不对可能是因为没有把Widget拉到最长。

xml内容提供者文件

该xml文件主要用于Widget的初始化和常规设置，如最小宽度和和高度，布局文件的设置，更新时间以及初始化图片等。根据要求修改即可。

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialKeyguardLayout="@layout/new_app_widget"
    android:initialLayout="@layout/new_app_widget"
    android:minHeight="50dp"
    android:minWidth="100dp"
    android:previewImage="@drawable/shoplist"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="86400000"
    android:widgetCategory="home_screen|keyguard"></appwidget-provider>
```

java文件

首先关于Widget的生命周期：

1. onEnabled方法：此方法在Widget第一次被创建的时候调用，并且只调用一次，此方法中常放入初始化数据，服务的操作。
2. onReceive方法：通BroadcastReceiver的OnReceive方法，但是这里有所不同的是，当接收到Widget操作时首先调用的是OnReceive方法，然后才是相关的操作方法。这也很好理解，Widget的是运行在桌面运用程序中的小控件，当自己的应用程序需要调用Widget是，就需要发送广播事件去调用。
3. onUpdate：Widget在固定的时间里更新时调用的方法。每次对Widget的更新以及固定周期系统都会对Widget调用该方法。
4. onDeleted：Widget被删除时调用的方法。
5. onDisabled：所用Widget被删除是调用的方法，同onEnabled方法相对。

这里我们只需要重写onReceive和onUpdate方法即可，一个用于接受广播时间的处理，一个用于定期更新Widget。Widget类中有一个静态方法updateAppWidget，用来后面更新Widget时调用。其中顶一个RemoteView 设置点击事件的监听器，然后再通过appWidgetManager进行更新。

```
static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                            int appWidgetId) {
    //CharSequence widgetText = context.getString(R.string.appwidget_text);
    // Construct the RemoteViews object
    RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.new_app_widget);
    views.setOnClickPendingIntent(R.id.mWidget, PendingIntent.getActivity(context,0,new Intent(context,MainActivity.class)
    // Instruct the widget manager to update the widget
    appWidgetManager.updateAppWidget(new ComponentName(context,NewAppWidget.class),views);
}
```

onUpdate方法是对系统中所有Widget进行更新，所以有一个appWidgetId的循环。事件的处理和后面的onReceive基本一致（写报告时先写的onReceive）。由于onReceive方法会在Widget建立时调用，所以这里intent中的跳转信息是跳到MainActivity。

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    // There may be multiple widgets active, so update all of them
    for (int appWidgetId : appWidgetIds) {
        updateAppWidget(context, appWidgetManager, appWidgetId);
        RemoteViews updateView = new RemoteViews(context.getPackageName(),R.layout.new_app_widget);
        Intent i = new Intent(context,MainActivity.class);
        PendingIntent pi = PendingIntent.getActivity(context,0, i ,PendingIntent.FLAG_UPDATE_CURRENT);
        updateView.setOnClickPendingIntent(R.id.mWidget,pi);
        ComponentName me = new ComponentName(context,NewAppWidget.class);
        appWidgetManager.updateAppWidget(me,updateView);
    }
}
```

重写onReceive方法，进行静态广播和动态广播的接收处理。
首先从广播的intent中取出对应good，然后需要用到一种用户程序访问主屏幕和修改特定区域内容的方法：RemoteView 架构。RemoteView 架构允许用户程序更新主屏幕的View，所以我们需要在接收到广播并获得一个随机商品信息后，利用RemoteView 对Widget进行更新。
首先定义RemoteView 的view，参数为当前上下文的包名和之前定义后的Widget布局。这里RemoteView 是用来更新Widget内容的。
然后创建Intent设置跳转到商品详情界面，并把接收到的广播内容即good添加到intent中。
接着定义pendingIntent设置事件触发即点击时的延迟intent，并将view 通过setTextView Text和setImageView Resource设置文本和图片，进行Widget的数据更新。然后设置监听器进行点击事件的处理。
最后通过AppWidgetManager调用updateAppWidget方法进行对Widget的数据更新。

```
@Override
public void onReceive(Context context, Intent intent){
    super.onReceive(context,intent);
```

```
    //AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
    //Bundle bundle = intent.getExtras();
    if(intent.getAction().equals(STATICACTION)){
        Goods good = (Goods) intent.getExtras().get("goods");
        // Construct the RemoteViews object
        RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.new_app_widget);
        Intent intent1 = new Intent(context,Goods_info.class);
        intent1.putExtras(intent.getExtras());
        PendingIntent localPendingIntent = PendingIntent.getActivity(context,0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
        views.setTextViewText(R.id.appwidget_text, good.getName()+"仅售"+good.getPrice()+"!");
        views.setImageViewResource(R.id.pic,good.getImageId());
        views.setOnClickPendingIntent(R.id.mWidget,localPendingIntent);
        ComponentName localComponentName = new ComponentName(context,NewAppWidget.class);
        AppWidgetManager.getInstance(context).updateAppWidget(localComponentName,views);

    }
    else if(intent.getAction().equals(DYNAMICACTION)){
        Goods good = (Goods) intent.getExtras().get("goods");
        // Construct the RemoteViews object
        RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.new_app_widget);
        Intent intent1 = new Intent(context,MainActivity.class);
        intent1.putExtras(intent.getExtras());
        PendingIntent localPendingIntent = PendingIntent.getActivity(context,0,intent1,PendingIntent.FLAG_UPDATE_CURRENT);
        views.setTextViewText(R.id.appwidget_text, good.getName()+"已添加到购物车");
        views.setImageViewResource(R.id.pic,good.getImageId());
        views.setOnClickPendingIntent(R.id.mWidget,localPendingIntent);
        ComponentName localComponentName = new ComponentName(context,NewAppWidget.class);
        AppWidgetManager.getInstance(context).updateAppWidget(localComponentName,views);
    }
}
```

动态广播的接受处理我也是在这里写的，和静态的基本一致，除了内容的区别。
然后两种广播的注册是在AndroidManifest文件里写的，在系统自己生成的基础上添加静态和动态的监听过滤。一开始我把动态广播的接受写在了MyStaticReceive即自定义的广播接收类中，但是我想如果都写在一起行不行，试了一下感觉可以，目前还没有发现什么问题。可能AndroidManifest中只能注册静态广播，但是AndroidManifest文件不会报错。

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

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/new_app_widget_info" />
</receiver>
```

其实这里的静态和动态广播的接受处理也可以写在自定义的MyStaticReceive类中，和通知的发出写在一起。

实验结果

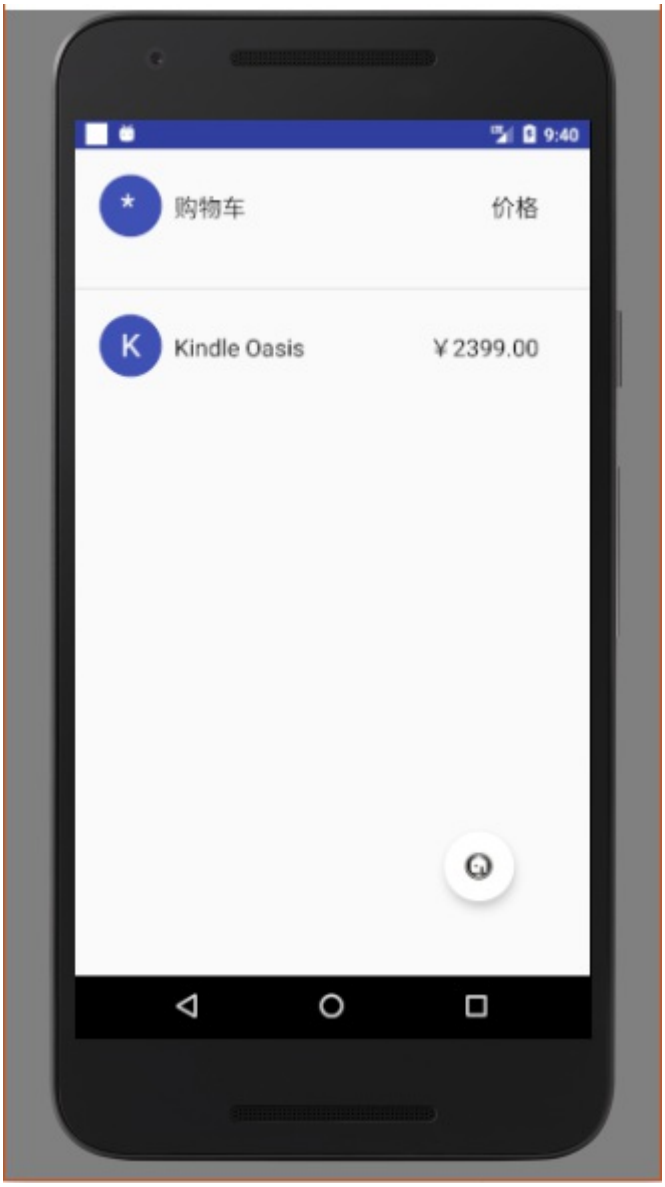
初始化：

点击w idget启动后由随机弹出的静态广播更新w idget：

点击w idget跳转到对应商品详情界面：

点击购物车图标添加到购物车后点击home键退出程序w idget更新：

点击w idget跳转到购物车界面：



实验总结

实验中遇到的问题：

1. 关于布局的问题。这里在写Widget的布局时遇到的问题是我一直想把图片贴在最左边，即margin-left=0，但是直接写图片和最左端还是有一定的距离，不会紧贴在最左端。因为使用的relative布局，使用layout_alignParentLeft=true也不行，但是后来两句都加上，把水平居中去掉之后跑了一次就可以了，但是我室友按照我的布局写他的图片还是不会贴在最左边，不知道具体是什么原因，一直觉得布局有的时候很玄学。
2. 数据的更新与跳转。这里想说的是由于我们的app并没有数据库来存储之前退出程序时的信息，所以如果按返回键退出程序后，此时点击已添加到购物车的widget并不会跳转到购物车并有相关信息，因为此时只是相当于启动程序，因为之前的操作已经在退出程序时销毁了。所以这里为了实现点击添加到购物车的widget跳转到购物车，要按中间的home键退出程序，此时程序还并未被销毁，即此时的购物车中有信息，并可以通过点击该widget跳转到购物车。所以之后的实验或者安卓开发中如果要实现点击返回键仍保留退出前的信息，则要加入数据库来保存信息，不令之前的操作信息被销毁。
3. RemoteViews的使用。由于我们不能直接对widget的数据如文字和图片直接进行修改，所以我们要利用RemoteViews实现对widget的数据更新，同时利用PendingIntent作为一个延迟的intent实现点击事件的跳转处理，否则直接利用intent无法实现在点击时才进行处理。
4. widget类中的updateAppWidget方法。该静态方法是对对应widget进行数据更新，我们在每次接收到广播进行事件处理时都要调用该方法，其实一开始并不清楚为什么要调用该方法，因为在onReceive和onUpdate方法中已经写了具体的数据更新操作，即通过RemoteViews和PendingIntent等进行的操作，但是只写了这些并无法完成对widget的表面的数据更新。也就是说，如果去掉manager调用该方法，那么虽然将某件商品添加到购物车后点击widget可以跳转到购物车并显示信息，但是widget仍然

是初始状态。所以这里我们必须利用manager调用该方法进行对widget的更新。就好像发出通知最后也要利用`manager.notify(0,notify);`实验通知的发出。所以在这里我认为所有这些自定义的根据操作做出对应响应的部件都要最后通过manager实现出来。

5. 还有一点的广播的接受和处理。这里我是把静态和动态的广播过滤监听都写在了AndroidManifest文件中，所以就直接在widget类中的onReceive方法中对两种广播进行了处理。但是实验文档中只把静态广播注册在了AndroidManifest中，然后动态广播在自定义的接受类中的onReceive方法中写。如果说不可以在AndroidManifest中注册动态广播的话那么只能按照实验文档中写，但是我这样写了之后目前没发现什么问题，所以就先这么写了。当然我也在MyStaticReceive的onReceive中写了对widget的动态广播的处理。

这次实验掌握了对安卓中屏幕小组件的创建和使用，虽然以前用安卓机时根本没在意过这个东西，但是在某些方面还是有一定作用的，比如音乐软件用来显示歌名和歌词等。widget的创建很简单，因为是AS中自带的类，直接右键添加就把布局，java等文件全都创建好了，我们只需要根据要求进行修改即可。widget也是通过接受广播进行数据的更新，所以由此看来广播的作用真的很大，程序内部，程序外部的通信大部分都要靠广播来完成。还有一点就是关于各种自定义组件的创建最后好像都要用到manager进行最后的建立，起码Notification和widget是这样。