Android移动开发基础案例教程

第6章 BroadcastReceiver（广播接收者）

- 广播接收者简介
- 广播接收者入门
- 自定义广播
- 广播的类型

作业点评

- 请简要说明SQLite数据库的创建过程
- 请简要说明BaseAdapter适配器包含几个抽象方法，以及这些方法的作用

预习检查

- 什么是广播接收者以及其作用
- 广播的类型

学习目标

3 广播接收者的创建

1 广播的类型

拦截有序广播

自定义广播 2

主讲内容

主讲内容

Speech content

广播接收者简介

发送广播消息

实际生活中，**电台**用于发送广播，**收音机**用于接收广播。

广播接收者简介

电台：中央人民广播电台，93.4mhz;
收音机：93.4mhz，收听广播;

Android系统内置的电台，发送一些事件：接收短信、外拨电话、电量不足、电量充满、SD插拔、软件安装与下载等。
Android应用程序中的收音机：广播接收者，指定需要接收的事件类型;
可以做一些对用户有用的业务逻辑操作。

接收应用发送的广播，可以是本应用也可以是其他应用，从而实现进程内或者进程间的通信，举个例子，如果你的手机安装了百度全家桶或者阿里全家桶或者企鹅全家桶等等这些豪华套餐，只要你启动其中一个应用，它就可以自定义一个广播然后发送出去，然后全家桶的其他成员接受到这个广播之后就可以全部在后台启动起来，干一些为人民服务的好事。

BroadcastReceiver广播接收者简介

BroadcastReceiver广播接收者是Android四大组件之一。
是一个全局监控组件
一般的都要在AndroidManifest中静态注册，但特殊的广播接收者也可以使用Java代码的方法来动态注册。
例：监听打电话的事件，在用户拨打电话的时候弹出一个toast

BroadcastReceiver广播接收者简介

BroadcastReceiver是用于接收广播的组件，用于组件与组件之间进行通信，可以跨应用程序传递.如操作系统电池电量低会发送一个广播，这样程序可以去监听这个广播，可以关闭程序里面比较耗电的功能，提示用户注意保存进度什么的，还如其它安装新应用等，还有普通应用程序，例如启动特定线程，文件下载完毕等。

广播接收者简介

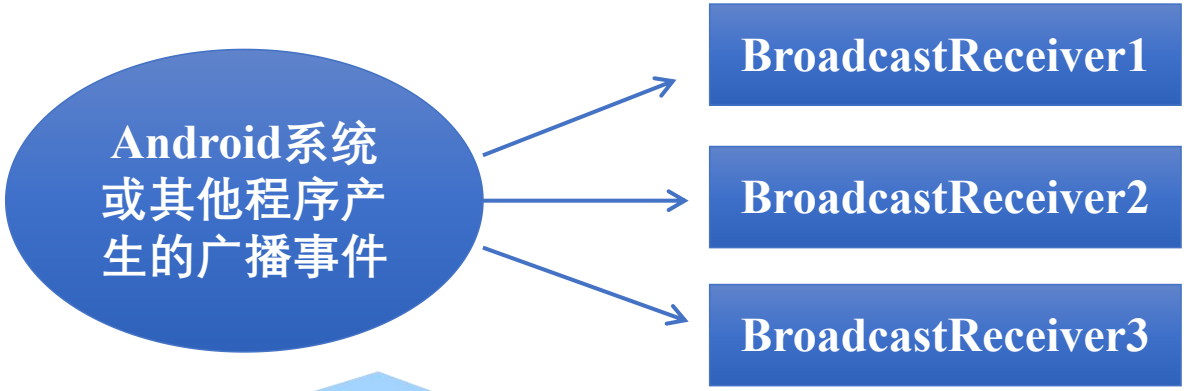
android中的广播机制设计的非常出色，很多事情原本需要开发者亲自操作的，现在只需等待广播告知自己就可以了，大大减少了开发的工作量和开发周期。

广播的发送的一般应用场合：发送频率低的情况可以使用，数据量小的情况可以使用。

广播接收者简介

广播特点

- Android系统中内置了很多广播，例如手机开机完成、电池电量不足时都会发送一条广播。
- 为了监听来自系统或者应用程序的广播事件，Android系统提供了BroadcastReceiver（广播接收者）组件。



当Android系统产生一个广播事件时，可以有多个对应的广播接收者接收并进行处理。

广播接收者简介

广播特点

1、即使广播接收者没有运行，当广播事件到达的时候，系统会自动启动广播接收者，并且调用onReceive方法处理消息；
2、从android4.0之后，添加了一个新特点，如果强制停止就相当于冻结了这个程序，只能等到下次手工启动后才能生效；

主讲内容

6.1 广播接收者简介

6.2 广播接收者入门

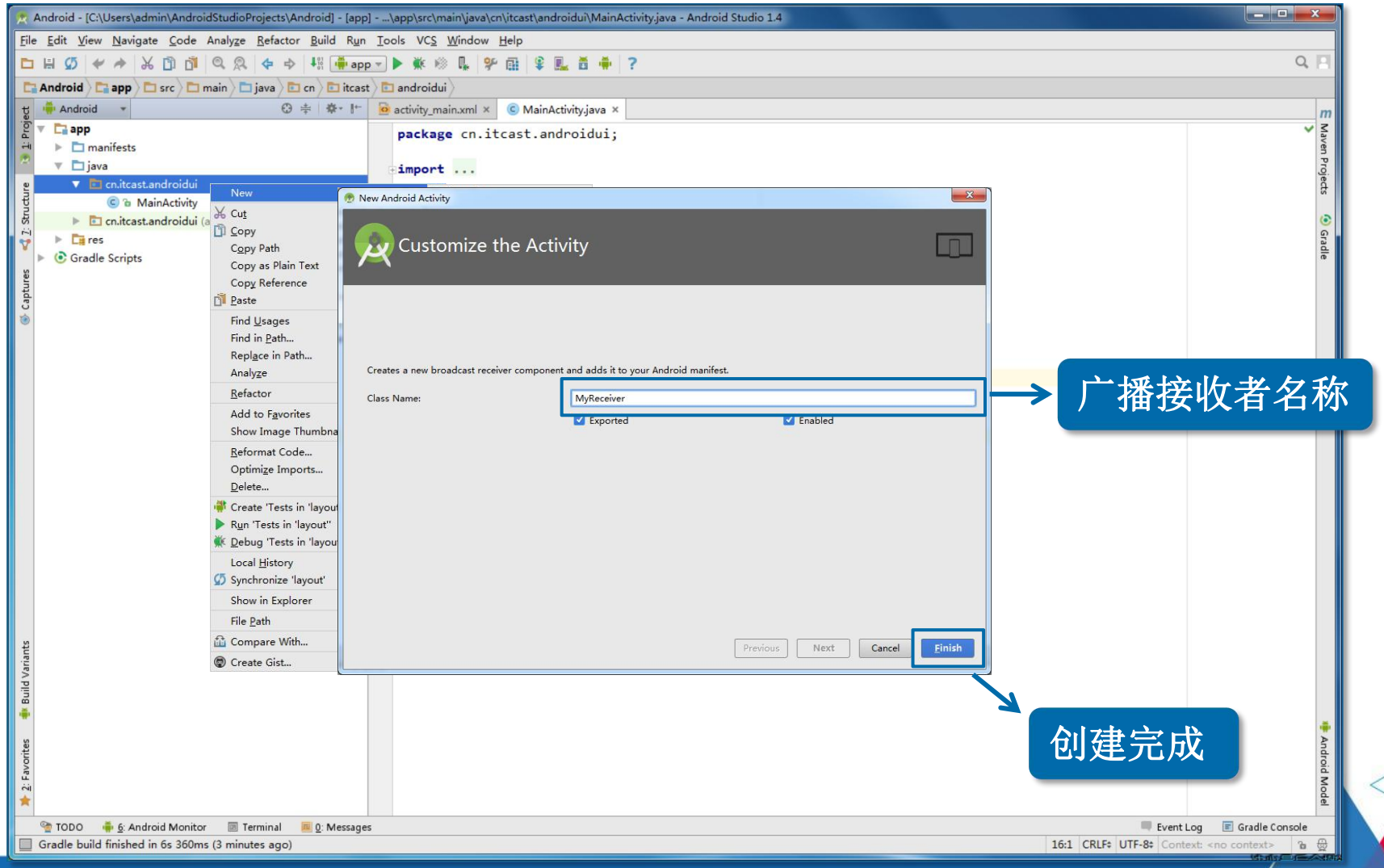
6.3 自定义广播

6.4 广播类型

主讲内容

Speech content

6.2.1 广播接收者的创建



广播接收者简介

要创建自己的BroadcastReceiver对象，需要继承BroadcastReceiver类，并实现其onReceive 方法。 在onReceive方法内，可以获取随广播而来的Intent中的数据，这非常重要，就像无线电一样，包含很多有用的信息。 在创建完BroadcastReceiver之后，还不能够使它进入工作状态，需要为它注册一个指定的广播地址。没有注册广播地址的BroadcastReceiver就像一个缺少选台按钮的收音机，虽然功能俱备，但也无法收到电台的信号。所以创建类后就要对BroadcastReceiver进行注册。

广播接收者简介

```
1  public class MyBroadcastReceiver extends BroadcastReceiver {
2      @Override
3      public void onReceive(Context context, Intent intent) {
4        //对接收到的广播进行处理，intent里面包含数据
5      }
6    }
```

Intent——意图，它可以带有数据，在Android设备上的任何应用程序组件间相互作用，将不同组件连在一起的桥梁。

Intent作用：

Intent最常用在启动新的Activity，不管是显式的（通过指定类来加载）还是隐式的（通过请求在一块数据上执行的动作）。

Intent还可以用来广播消息，然后使用Broadcast Receiver来监听并响应这些Intent。

使用Intent来传播动作，比如通过我们的程序中打电话。

BroadcastReceiver注册

1.静态注册

静态注册是在AndroidManifest.xml文件中配置的

```
1  <receiver android:name="com.hua.bcreceiver.MyBroadcastReceiver" >
2 <intent-filter>
3   <action android:name="android.intent.action.MY_BROADCAST" />
4   <category android:name="android.intent.category.DEFAULT" />
5  </intent-filter>
6        </receiver>
```

Intent-filter用来注册Activity、Service、Broadcast Receiver具有在某种数据上处理一个动作的能力。

当一个应用程序为了向Android说明它有为某种数据或者其他应用程序组件提供处理动作的能力时，它需要在manifest中注册为一个Intent处理者。

BroadcastReceiver注册

动态注册(不是常驻型广播)

动态注册需要在代码中动态的指定广播地址并注册，注意，registerReceiver是android.content.ContextWrapper类中的方法，Activity 和Service都继承了ContextWrapper，所以可以直接调用。在实际应用中，在 Activity或Service中注册了一个BroadcastReceiver，当这个Activity或Service被销毁时 如果没有解除注册，系统会报一个异常，提示是否忘记解除注册了。所以，记得在 特定的地方执行解除注册操作。

注册代码

```
MyBroadcastReceiver mbcr = new MyBroadcastReceiver(); IntentFilter filter = new IntentFilter();
registerReceiver(mbcr, filter);// 注册
（2）解除注册代码，一般在页面销毁时操作
@Override
protected void onDestroy() {
        unregisterReceiver(mbcr);
        mbcr = null;
        super.onDestroy();
    }
```

广播的registerReceiver() 和 unregisterReceiver()要成对出现

如果 registerReceiver()和 unregisterReceiver()不成对出现，则可能导致已经注册的receiver 没有在合适的时机注销，导致内存泄漏，占用内存空间，加重 SystemService负担。部分华为的机型会对 receiver 进行资源管控，单个应用注册过多 receiver 会触发管控模块抛出异常，应用直接崩溃。

广播接收者简介

通过上面两步其实已经完成了一个简单功能的广播接收者，已经可以具有接受广播功能了。在onReceive方法里面就可以做自己想要的处理。

但是上面的BroadcastReceiver注册时的action动作意图值是自己定义的字符串，需要自己发送对应意图的广播才能让上面的广播接收到消息，如果是上面使用系统定义好的动作意图action，那么，就可以接收到系统发过来的广播信息，比如手机接收到短信，手机底层是会发送一条广播，查看手机系统内是否有符合条件的广播接收者，如果有就会向它发送信息。。。

静态注册的广播的程序运行一次后，以后能一直保持广播接收的状态，只要有对应的广播就能接收它的广播信息。而动态注册的广播需要程序在运行中才能接收广播。

关于优先级的说明

在静态代码块中定义：

```
<intent-filter
android:priority="999">
```

使用动态java代码设置：

```
filter.setPriority(999);
```

优先级的值越大它的优先级就越高。

那么priority的最大值是多少呢？

网上或有些书说的是1000，但是都不正确，正确答案是2147483647。优先级的值比它大1都会报错。并且优先级1001确实比优先级1000更快接收到广播。

优先级的范围是：-2147483647到2147483647

刚好它的长度范围是Integer类型数据的最大值4294967296（2的32次方）。

二．发送广播

上面在广播定义好的情况下，可以接收系统的广播信息，也可以接收自己发送的广播信息。

问题来了：

如果有多个接收者都注册了相同的广播地址，又会是什么情况呢，能同时接收到同一条广播吗，相互之间会不会有干扰呢？这就涉及到普通广播和有序广播的概念了。

（一）普通广播

普通广播对于多个接收者来说是完全异步的，通常每个接收者都无需等待即可以接收到广播，接收者相互之间不会有影响。对于这种广播，接收者无法终止广播，即无法阻止其他接收者的 接收动作。

主要代码：

```
Intent intent = new Intent( "myBroadcastReceiver" );
intent.putExtra( "msg" , "这是MainActivity页面发送的广播—-》》" );
sendBroadcast(intent);
```

（二）有序广播

有序广播比较特殊，它每次只发送到优先级较高的接收者那里，然后由优先级高的接受者再传播到优先级低的接收者那里，优先级高的接收者有能力终止这个广播。可以在广播注册时使用intent-filter里面的android: priority=”xxx”去解决或在java代码中用setPriority（xxx）来设置。

主要代码：

```
Intent intent = new Intent( "myBroadcastReceiver" );
intent.putExtra( "msg" , "这是MainActivity页面发送的广播--》》" );
sendOrderedBroadcast(intent,null);//如果是该自定义的广播接收者发送广播，第二个参数一般为null
```

对比有序广播和无序广播：

发现他们的使用方式没什么不同，只是有序广播比无序广播要多一个参数;

但是它们的作用是非常不同的，无序广播是不可被截断的，每一个广播接收者都可以收到这个无序广播发送的广播信息；而无序广播时候根据优先级数的大小来判断谁先接收到广播信息，并且可以选择是否截断这个广播信息，如果在某个广播接收者截断信息，在它的优先级之下的广播接收者都接收不到广播信息。

拦截方法： abortBroadcast();

有序广播才能拦截。

三．自定义广播接收者和广播发送的示例

这里的程序是在两个程序中的，但是也可以写在多个程序中，或单个程序中。程序的效果是一样的。因为BroadcastReceiver任何时候都可以接收广播消息，而发送广播消息，只要有上下文的地方都可以。

（一）程序一的设计

静态广播接收者的接收测试

1.创建第一个广播接收者类

```
1  package com.lwz.receiver;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.util.Log;
7  import android.widget.Toast;
8
9  /**
10  * 广播接收者的使用
11  * 这里使用的是静态注册的方法，等待接收广播
12  */
13
```

```
14 public class MyBroadcastReceiver extends BroadcastReceiver {
15     @Override
16     public void onReceive(Context context, Intent intent) {
17         Log.e("Receiver", "这是静态的广播接受者（优先级500）--->" + intent.getStringExtra("msg"));
18         Toast.makeText(context, "这是使用静态的方法来创建的广播（优先级500）接受者收到消息", Toast.LENGTI
19         abortBroadcast();
20     }
21 }
```

Toast.makeText()各个参数:

Toast:是一个类，主要管理消息的提示。

makeText()，是Toast的一个方法，用来显示信息，分别有三个参数。

第一个参数：this，是上下文参数，指当前页面显示

第二个参数：“string string string ”是你想要显示的内容，也可以是“你好！”。这个是随便定义的，显示你想要显示的内容。

第三个参数：Toast.LENGTH_LONG，是你指你提示消息，显示的时间，这个是稍微长点儿，对应的另一个是ToastLENGTH_SHORT，这个时间短点儿，大概2秒钟。

show()，表示显示这个Toast消息提醒，当程序运行到这里的时候，就会显示出来，如果不调用show()方法，这个Toast对象存在，但是并不会显示，所以一定不要忘记。

2.创建第二个广播接收者类

```
1  package com.lwz.receiver;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.util.Log;
7  import android.widget.Toast;
8
9  /**
10  * 广播接收者的使用
11  * 这里使用的是静态注册的方法，等待接收广播
12  */
13
```

```
14  public class MyBroadcastReceiver2 extends BroadcastReceiver {
15
16
17      @Override
18      public void onReceive(Context context, Intent intent) {
19
20          Log.e("Receiver", "这是静态的广播接受者（优先级999）---》" + intent.getStringExtra("msg"));
21          Toast.makeText(context, "这是使用静态的方法来创建的广播接受者（优先级999）收到消息", Toast.LENGTI
22          //截断广播
23          abortBroadcast();
24          //报错 BroadcastReceiver trying to return result during a non-ordered broadcast
25          //好像是要有序广播才能截断
26
27      }
28
29
30  }
```

3.在AndroidManifest中注册这两个广播接收者

```
<receiver android:name=".MyBroadcastReceiver">
            <intent-filter android:priority="500">
                <action android:name="myBroadcastReceiver" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>
        <receiver android:name=".MyBroadcastReceiver2">
            <intent-filter android:priority="999">
                <action android:name="myBroadcastReceiver" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>
```

注意这里的广播接收者1和2，基本数据是相同的，但是优先级不同。

4.布局文件main_activity.xml的设计

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6
7      <TextView
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         android:layout_gravity="center_horizontal"
11         android:text="广播的发送" />
12
```

```
13      <Button
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:onClick="sendToSXL"
17          android:text="给静态注册的广播发送信息" />
18
19      <Button
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:onClick="sendToJava"
23          android:text="给动态注册的广播发送信息" />
24
25      <Button
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:onClick="sendOrderToXML"
29          android:text="给静态注册的广播接收者发送有序广播信息" />
30
```

```
31      <Button
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:onClick="sendOrderToJava"
35          android:text="给动态注册的广播接收者发送有序广播信息" />
36
37  </LinearLayout>
```

5.MainActivity.java文件的设计代码

```java
package com.lwz.receiver;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

/**
 * 本类用来发送广播
 */
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

```
18
19      //给静态广播发送信息
20      public void sendToSXL(View view) {
21          //发送广播使用sendBroadcast，需要指定Intent的action来说明发送什么样的广播。
22          //Intent里面的参数是action，要和静态注册的广播的actin对应
23          Intent intent = new Intent("myBroadcastReceiver");
24          intent.putExtra("msg", "这是MainActivity页面发送的广播---》》");
25          sendBroadcast(intent);
26
27      }
28
29      //给动态广播发送信息
30      public void sendToJava(View view) {
31          //发送广播使用sendBroadcast，需要指定Intent的action来说明发送什么样的广播。
32          //Intent里面的参数是action，要和动态注册的广播的actin对应
33          Intent intent = new Intent("javaReceiver");
34          intent.putExtra("msg", "这是MainActivity页面发送的广播---》》");
35          sendBroadcast(intent);
36      }
```

```
37
38      //给静态注册的广播发送有序广播信息
39      public void sendOrderToXML(View view) {
40          //发送有序广播使用sendOrderedBroadcast，需要指定Intent的action来说明发送什么样的广播。
41          //Intent里面的参数是action，要和静态注册的广播的actin对应
42          Intent intent = new Intent("myBroadcastReceiver");
43          intent.putExtra("msg", "这是MainActivity页面发送的广播----》》");
44          sendOrderedBroadcast(intent,null);
45
46      }
47      //给静态注册的广播发送有序广播信息
48      public void sendOrderToJava(View view) {
49          //发送广播使用sendOrderedBroadcast，需要指定Intent的action来说明发送什么样的广播。
50          //Intent里面的参数是action，要和静态注册的广播的actin对应
51          Intent intent = new Intent("javaReceiver");
52          intent.putExtra("msg", "这是MainActivity页面发送的广播----》》");
53          sendOrderedBroadcast(intent,null);
54
55      }
56
57
58
59  }
```

上面主要负责广播发送代码的设计。如果要发送的广播是没有对应的广播接收者，程序不会报错，只是会没有反应而已。

上面程序其实已经可以运行了，可以对静态的广播接收者发送广播。

如果是发送无序广播，两个广播接收者都会有消息，但是会出现错误报告！告诉你无序广播不能截断。

如果是发送有序广播，那么低优先级的广播不能接收广播数据。因为这里使用了拦截的方法!

（二）程序二的设计：

动态广播接收者的测试

1.第一个广播接收者的创建

```
1
2  package com.lwz.receiver2;
3
4  import android.content.BroadcastReceiver;
5  import android.content.Context;
6  import android.content.Intent;
7  import android.util.Log;
8  import android.widget.Toast;
9
10 /**
11  * 这里使用动态的方法来创建一个广播接受者
12  * 动态的广播接收者必须在程序启动后才能接收广播，静态的广播不需要启动就可以接收!
13  */
```

```
11  * 这里使用动态的方法来创建一个广播接受者
12  * 动态的广播接收者必须在程序启动后才能接收广播，静态的广播不需要启动就可以接收!
13  */
14
15 public class BroadcastReceiveForjava extends BroadcastReceiver {
16     @Override
17     public void onReceive(Context context, Intent intent) {
18         Log.e("Receiver", "这是使用动态的方法来创建的广播接受者1)----》" + intent.getStringExtra("msg"
19         Toast.makeText(context, "这是使用动态的方法来创建的广播接受者1收到消息", Toast.LENGTH_SHORT).sh
20        abortBroadcast();
21     }
22 }
```

2.第二个广播接收者的创建

```
package com.lwz.receiver2;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;
```

```
/**
* 这里使用动态的方法来创建一个广播接受者
* 动态的广播接收者必须在程序启动后才能接收广播，静态的广播不需要启动就可以接收！
*/

public class BroadcastReceiveForjava2 extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
Log.e("Receiver", "这是使用动态的方法来创建的广播接受者2—-》" + intent.getStringExtra("msg"));
Toast.makeText(context, "这是使用动态的方法来创建的广播接受者2收到消息", Toast.LENGTH_SHORT).show();
abortBroadcast();
}
}
```

3.动态注册上面两个广播接收者

```
1  package com.lwz.receiver2;
2
3  import android.content.IntentFilter;
4  import android.os.Bundle;
5  import android.support.v7.app.AppCompatActivity;
6
7  /**
8   * 这里使用动态的方法来创建一个广播接受者
9   * 动态的广播接收者必须在程序启动后才能接收广播，静态的广播不需要启动就可以接收!
10  * 动态广播的类创建之后要去使用代码注册才能生效
11  * 另外动态注册后要在某一个时间去解除绑定，否者会报错!
12  */
13 public class MainActivity extends AppCompatActivity {
14
15     //定义广播接收者
16     BroadcastReceiveForjava receiver;
17     BroadcastReceiveForjava2 receiver2;
18
```

```
19      @Override
20      protected void onCreate(Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22          setContentView(R.layout.activity_main);
23          //创建广播接收者对象
24          receiver = new BroadcastReceiveForjava();
25          receiver2 = new BroadcastReceiveForjava2();
26          //注册广播接收者,需要一个意图对象,也需要action参数,这里是定义Action参数
27          IntentFilter filter = new IntentFilter("javaReceiver");
28          IntentFilter filter2 = new IntentFilter("javaReceiver");
29          //动态设置广播的优先级
30          filter.setPriority(999);
31          filter2.setPriority(10);
32          //注册广播,
33          // 动态广播拦截?先注册者先收到消息???
34          registerReceiver(receiver2, filter);
35          registerReceiver(receiver, filter);
36      }
```

```
37
38      @Override
39      protected void onDestroy() {
40          //在适当的时候要解除广播接收者的绑定
41          unregisterReceiver(receiver);
42          unregisterReceiver(receiver2);
43          super.onDestroy();
44      }
45  }
```

程序二运行后，界面并没有画面。因为布局文件并没有设计。

这时不要退出程序二，点击回到主页，就可以再次进入程序一的界面，验证广播的接收和发送的效果。

程序运行后：

点击第一个按钮，对静态注册的广播接收者发送的无序广播，Log信息如下：

```
E/Receiver: 这是静态的广播接受者（优先级500）---》这是MainActivity页面发送的广播----》》
E/BroadcastReceiver: BroadcastReceiver trying to return result during a non-ordered broadcast
```

```
/Receiver: 这是静态的广播接受者（优先级999）---》这是MainActivity页面发送的广播----》》
/BroadcastReceiver: BroadcastReceiver trying to return result during a non-ordered broadcast
```

可以看到两个广播接收者都收到能收到广播的信息，只是报了一个错误给你！告诉你不能拦截无序的广播。

点击第三个按钮，对静态注册的广播接收者发送有序广播，Log信息如下：

```
writing handshake bytes: Broken pipe (-1 of 1)
E/Receiver: 这是静态的广播接受者（优先级999）---》这是MainActivity页面发送的广播----》》
```

可以看到成有序广播可以被截断，低等级的广播接收者不能收到被截断的广播。

点击第二个按钮，对动态注册的广播接收者发送无序广播，Log信息如下：

```
E/Receiver: 这是使用动态的方法来创建的广播接受者2----》这是MainActivity页面发送的广播----》》
E/BroadcastReceiver: BroadcastReceiver trying to return result during a non-ordered broadcast
```

```
r2 E/Receiver: 这是使用动态的方法来创建的广播接受者1)----》这是MainActivity页面发送的广播----》》
r2 E/BroadcastReceiver: BroadcastReceiver trying to return result during a non-ordered broadcast
```

可以看到两个动态注册的广播接收者都可以接收到这个无序的广播。但是也是有报错！告诉你不能拦截无序广播哦。

点击第四个按钮，对动态注册的广播接收者发送有序广播，Log的信息如下：

可以看到有序广播可以被优先级高的动态广播接收者截断。

后面可以改变优先级来看看优先级是否起到作用！

从上面程序说明：无序广播不能截断，有序广播能被高优先级的广播接收者截断。不管是对静态的广播接收者还是对动态的广播接收者都是一样的。

四．接收手机短信广播信息的一个示例

（一）创建广播接收者

```
1  package com.lwz.phoneData;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.telephony.SmsMessage;
7  import android.util.Log;
8  import android.widget.Toast;
9
10 import java.text.SimpleDateFormat;
11 import java.util.Date;
12
13 /**
14  * 创建一个短信拦截广播接收者
15  */
```

```
16
17  public class SMSBroadcastReceiver extends BroadcastReceiver {
18      private static final String action = "android.provider.Telephony.SMS_RECEIVED";
19
20      @Override
21      public void onReceive(Context context, Intent intent) {
22          if (intent.getAction().equals(action)) {
23              Object[] pduses = (Object[]) intent.getExtras().get("pdus");
24              String mobile = "";
25              String content = "";
26              String time = "";
27              System.out.println(pduses.length);
28              for (Object pdus : pduses) {
29                  byte[] pdusmessage = (byte[]) pdus;
30                  SmsMessage sms = SmsMessage.createFromPdu(pdusmessage);
31                  mobile = sms.getOriginatingAddress();// 发送短信的手机号码
32                  content = sms.getMessageBody(); // 短信内容
33                  Date date = new Date(sms.getTimestampMillis());// 发送日                期
34                  SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
35                  time = format.format(date); // 得到发送时间
36              }
```

```
            Toast.makeText(context, context.getResources().getString(R.string.app_name), Toast.LEN
            String show = "发送者:" + mobile + " 内容:" + content + "      时间:" + time;
            Toast.makeText(context, show, Toast.LENGTH_LONG).show();
        }
        Log.e("TAG","拦截短信");
        abortBroadcast();
    }

}
```

（二）在AndroidManifest中注册

```
<receiver android:name=".SMSBroadcastReceiver">
            <intent-filter android:priority="1001">
              <action android:name="android.provider.Telephony.SMS_RECEIVED" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>

这里action的属性值必须正确，否则不能监听到这个广播！
这里还要添加手机权限：
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

五．系统广播定义的一些action值展示：

1.Intent.ACTION_AIRPLANE_M;

关闭或打开飞行模式时的广播

2.Intent.ACTION_BATTERY_CH;

（1）充电状态，或者电池的电量发生变化;

（2）电池的充电状态、电荷级别改变，不能通过组建声;

3.Intent.ACTION_AIRPLANE_MODE_CHANGED;

关闭或打开飞行模式时的广播

4.Intent.ACTION_BATTERY_CHANGED;

（1）充电状态，或者电池的电量发生变化

（2）电池的充电状态、电荷级别改变，不能通过组建声明 接收这个广播，只有通过registerReceiver()注册

5.Intent.ACTION_BATTERY_LOW;

表示电池电量低

6 .Intent.ACTION_BATTERY_OKAY;

表示电池电量充足，即从电池电量低变化到饱满时会发出广播

7.Intent.ACTION_BOOT_COMPLETED;

在系统启动完成后的广播，这个动作被广播一次（只有一次）。

8.Intent.ACTION_CAMERA_BUTTON;

按下照相时的拍照按键(硬件按键)时发出的广播

9.Intent.ACTION_CLOSE_SYSTEM_DIALOGS;

当屏幕超时进行锁屏时,当用户按下电源按钮,长按或短按(不管有没跳出话框)，进行锁屏时,android系统都会广播此Action消息

10.Intent.ACTION_CONFIGURATION_CHANGED;

设备当前设置被改变时发出的广播(包括的改变:界面语言，设备方向，等，请参考Configuration.java)

11.Intent.ACTION_DATE_CHANGED;

设备日期发生改变时会发出此广播

12.Intent.ACTION_DEVICE_STORAGE_LOW;

设备内存不足时发出的广播,此广播只能由系统使用，其它APP不可用

13.Intent.ACTION_DEVICE_STORAGE_OK;

设备内存从不足到充足时发出的广播,此广播　　只能由系统使用，其它APP不可用

14.Intent.ACTION_DOCK_EVENT;

发出此广播的地方

frameworks\base\services\java\com\android\server\DockObserver.java

15.Intent.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE;

移动APP完成之后，发出的广播(移动是指:APP2SD)

16.Intent.ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE;

正在移动APP时，发出的广播(移动是指:APP2SD) 17.

17.Intent.ACTION_GTALK_SERVICE_CONNECTED;

Gtalk已建立连接时发出的广播

18.Intent.ACTION_GTALK_SERVICE_DISCONNECTED;

Gtalk已断开连接时发出的广播

19.Intent.ACTION_HEADSET_PLUG;

在耳机口上插入耳机时发出的广播

20.Intent.ACTION_INPUT_METHOD_CHANGED;

改变输入法时发出的广播

21.Intent.ACTION_LOCALE_CHANGED;

设备当前区域设置已更改时发出的广播

22.Intent.ACTION_MANAGE_PACKAGE_STORAGE;

表示用户和包管理所承认的低内存状态通知应该开始。

23. Intent.ACTION_MEDIA_BAD_REMOVAL;

未正确移除SD卡(正确移除SD卡的方法:设置–SD卡和设备内存–卸载SD卡)，但已把SD卡取出来时发出的广播,扩展介质（扩展卡）已经从 SD 卡插槽拔出，但是挂载点 (mountpoint) 还没解除 (unmount)

24.Intent.ACTION_MEDIA_BUTTON;

按下” Media Button” 按键时发出的广播,假如有” Media Button” 按键的话(硬件按键)

25. Intent.ACTION_MEDIA_CHECKING;

插入外部储存装置，比如SD卡时，系统会检验SD卡，此时发出的广播。

26.Intent.ACTION_MEDIA_EJECT;

已拔掉外部大容量储存设备发出的广播（比如SD卡，或移动硬盘）,不管有没有正确卸载都会发出此广播, 用户想要移除扩展介质（拔掉扩展卡）。

27.Intent.ACTION_MEDIA_MOUNTED;

插入SD卡并且已正确安装（识别）时发出的广播, 扩展介质被插入，而且已经被挂载。

28.Intent.ACTION_MEDIA_NOFS;

拓展介质存在，但使用不兼容FS（或为空）的路径安装点检查介质包含在Intent.mData领域。

29. Intent.ACTION_MEDIA_REMOVED;

外部储存设备已被移除，不管有没正确卸载,都会发出此广播， 扩展介质被移除。

30.Intent.ACTION_MEDIA_SCANNER_FINISHED;

广播：已经扫描完介质的一个目录

31.Intent.ACTION_MEDIA_SCANNER_SCAN_FILE;

请求媒体扫描仪扫描文件并将其添加到媒体**数据库**。

32.Intent.ACTION_MEDIA_SCANNER_STARTED;

广播：开始扫描介质的一个目录

33. Intent.ACTION_MEDIA_SHARED;

广播：扩展介质的挂载被解除 (unmount)，因为它已经作为 USB 大容量存储被共享。

广播：扩展介质的挂载被解除 (unmount)，因为它已经作为 USB 大容量存储被共享。

34.Intent.ACTION_MEDIA_UNMOUNTABLE;

35.Intent.ACTION_MEDIA_UNMOUNTED

广播：扩展介质存在，但是还没有被挂载 (mount)

36.Intent.ACTION_NEW_OUTGOING_CALL;

37. Intent.ACTION_PACKAGE_ADDED;

（1）成功的安装APK之后

（2）广播：设备上新安装了一个应用程序包。

（3）一个新应用包已经安装在设备上，数据包括包名（最新安装的包程序不能接收到这个广播）

38.Intent.ACTION_PACKAGE_CHANGED;

一个已存在的应用程序包已经改变，包括包名

39.Intent.ACTION_PACKAGE_DATA_CLEARED;

（1）清除一个应用程序的数据时发出的广播(在设置 - - 应用管理 - - 选中某个应用，之后点清除数据时?)

（2）用户已经清除一个包的数据，包括包名（清除包程序不能接收到这个广播）

40.Intent.ACTION_PACKAGE_INSTALL;

触发一个下载并且完成安装时发出的广播，比如在电子市场里下载应用？

41.Intent.ACTION_PACKAGE_REMOVED;

成功的删除某个APK之后发出的广播，一个已存在的应用程序包已经从设备上移除，包括包名（正在被安装的包程序不能接收到这个广播）

42.Intent.ACTION_PACKAGE_REPLACED;

替换一个现有的安装包时发出的广播（不管现在安装的APP比之前的新还是旧，都会发出此广播？）

43.Intent.ACTION_PACKAGE_RESTARTED;

用户重新开始一个包，包的所有进程将被杀死，所有与其联系的运行时间状态应该被移除，包括包名（重新开始包程序不能接收到这个广播）

44.Intent.ACTION_POWER_CONNECTED;

插上外部电源时发出的广播

45.Intent.ACTION_POWER_DISCONNECTED;

已断开外部电源连接时发出的广播

46.Intent.ACTION_PROVIDER_CHANGED;

47.Intent.ACTION_REBOOT;

重启设备时的广播

48.Intent.ACTION_SCREEN_OFF;

屏幕被关闭之后的广播

49. Intent.ACTION_SCREEN_ON;

屏幕被打开之后的广播

50.Intent.ACTION_SHUTDOWN;

关闭系统时发出的广播

51. Intent.ACTION_TIMEZONE_CHANGED;

时区发生改变时发出的广播

52.Intent.ACTION_TIME_CHANGED;

时间被设置时发出的广播

53.Intent.ACTION_TIME_TICK;

53.Intent.ACTION_TIME_TICK;

广播：当前时间已经变化（正常的时间流逝），当前时间改变，每分钟都发送，不能通过组件声明来接收，只有通过Context.registerReceiver()方法来注册

54. Intent.ACTION_UID_REMOVED;

一个用户ID已经从系统中移除发出的广播

55. Intent.ACTION_UMS_CONNECTED;

设备已进入USB大容量储存状态时发出的广播？

56.Intent.ACTION_UMS_DISCONNECTED;

设备已从USB大容量储存状态转为正常状态时发出的广播？

57.Intent.ACTION_USER_PRESENT;

58.Intent.ACTION_WALLPAPER_CHANGED;

设备墙纸已改变时发出的广播

六．广播接收者BroadcastReceiver总结

发送频率低的可以使用 数据量小在可使用 intent携带的数据不能太大 广播的操作一定要在5秒钟内完成（10秒不能完成，ANR），如果耗时时间太长，开启服务。

（一）两种注册方法

1. 静态注册，manifest文件中注册,广播应该是一个外部类

优点：程序不启动，一样能收到广播，一般用于推送(自己搭建)。系统的处理。

缺点：占用资源较多。只有广播，系统会去匹配所有的接受者。

2.动态注册 一般都是匿名类代码中通过Content的方法registerReceiver（receiver，intentfilter）注册广播； 在退出的时候一定需要解绑，否则抛异常。unRegisterReceiver 用于程序组件之间的消息传递。需要刷新UI

优点：程序占用资源少，及时释放

缺点：一定需要组件启动，注册广播。

（二）发送广播

两种发送方法

1.无序广播 消息同时到达，不能延迟 sendBroadcast(Intent);

2. 有序广播

消息按照优先级来传递，可以延迟，并且可以拦截 sendOrderedBroadcast(Intent,permission); permission可以为null，表明不需要额外的权限,如果不为空，所有接收方必须有这个权限。

拦截方法 abortBroadcast();

广播优先级，应该是有序广播才能有效果;

广播最大的作用 Activity与Service进行通信 MyApp通知所有的界面。

其中广播的静态广播和动态广播，有序广播和无序广播。这个需要有比较清楚的思路。

静态广播和动态广播指的是广播接受者注册方式，静态广播接收者是在AndroidManifest中用xml语句设置；动态广播接收者是在某个java文件中使用java代码注册。

有序广播和无序广播指的是发送广播的方式：根据广播接收者的优先级来确定哪一个广播接收者先获得广播，无序广播不能被拦截，有序广播会可以被拦截。

所以说动态和静态是对广播接收者的描述；

有序和无序是是对发送的广播本身的描述。

6.2.2 实战演练——拦截史迪仔电话

1 功能描述： 实现拦截保存到手机中的号码。

2 技术要点： BroadcastReceiver

3 实现步骤：

① 用户交互界面的设计与实现

② 界面交互代码的设计与实现

③ 创建广播接收者OutCallReceiver.java

④ 注册广播接收者

案例代码（详见教材P21—P26）

6.3.2 实战演练——拯救史迪仔

1 功能描述： 接收一条自定义的广播。

2 技术要点： 发送一条自定义的广播，并创建广播类实现接收

3 实现步骤：

① 用户交互界面的设计与实现

② 界面交互代码的设计与实现

③ 创建广播接收者MyBroadcastReceiver.java

④ 在清单文件中设置自定义广播接收者的事件类型

案例代码（详见教材P21—P26）

6.4.2 实战演练——拦截史迪仔广播

1 功能描述： 实现拦截一条有序广播。

2 技术要点： 通过sendOrderedBroadcast()方法发送一条有序广播

3 实现步骤：

① 用户交互界面的设计与实现

② 界面交互代码的设计与实现

③ 创建3个广播接收者：

MyBroadcastReceiverOne.java

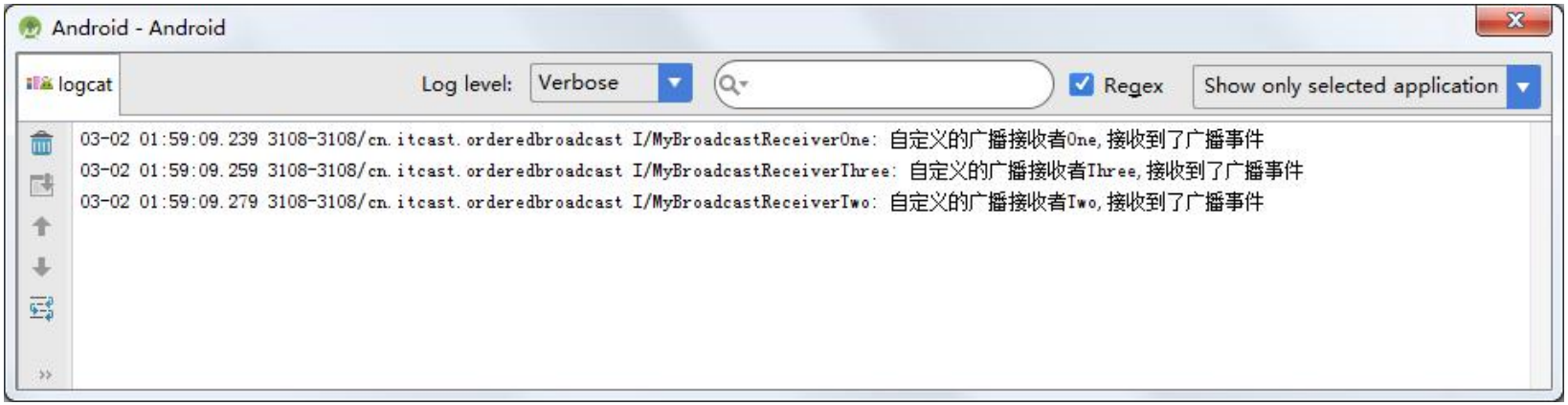
MyBroadcastReceiverTwo.java

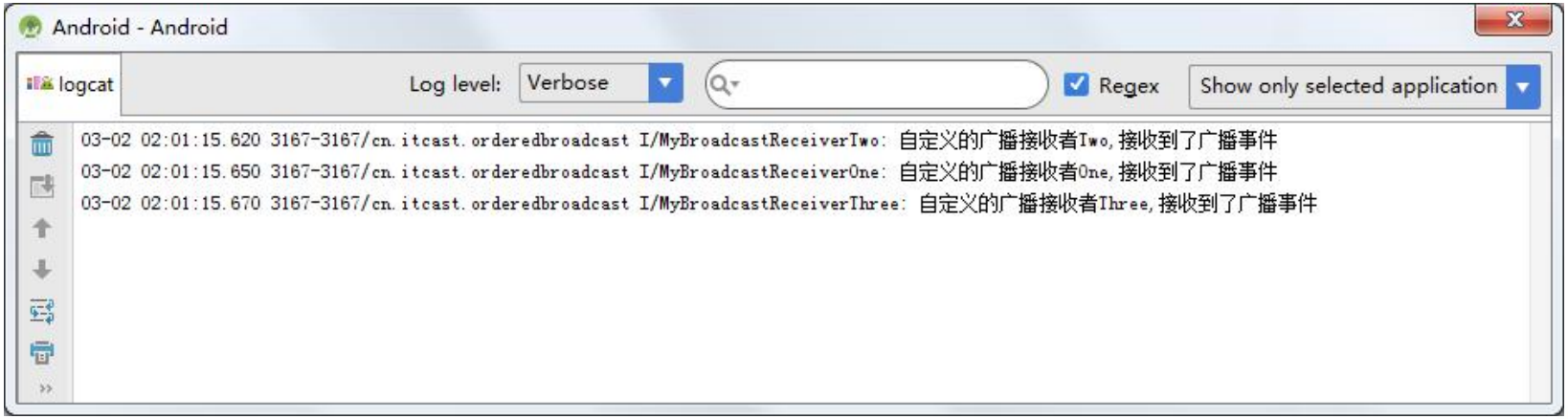
MyBroadcastReceiverThree.java

④ 设置优先级广播接收者的优先级

案例代码（详见教材P21—P26）

6.4.2 实战演练——拦截史迪仔广播





6.5 本章小结

本章详细地讲解了广播接收者的相关知识，首先介绍了什么是广播接收者，然后讲解了如何自定义广播以及广播的类型。通过本章的学习，要求初学者能够熟练掌握广播接收者的使用，并在实际开发中进行应用。

本章作业

- 请简要说明要拦截外拨电话需要使用什么权限。
- 请简要说明注册广播有几种方式，以及每种方式的特点。

预习作业

- 服务分为几种
- 服务的生命周期

Thank You!