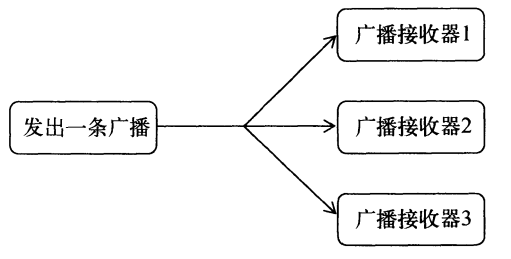
**第五章 详解广播机制**

1. **广播机制简介**

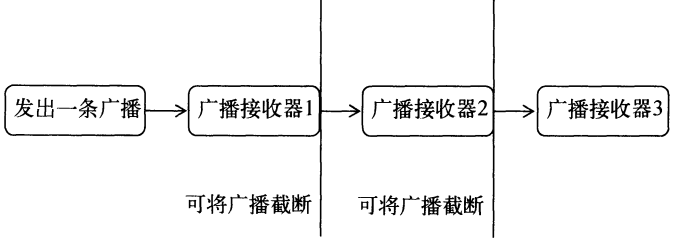
**Android中的广播机制更加灵活，因为每个应用程序都可以对自己感兴趣的广播进行注册并只接受它们。发送广播需要借助Intent，接收广播则需要广播接收器（Broadcast Receiver）。**

**Android中的广播类型分为两种：**

1. **标准广播 在广播发出之后, 所有的广接收器几乎都会在同一时刻接收到这条广播消息，没有先后顺序，效率高且无法被截断。工作示意图如下：**



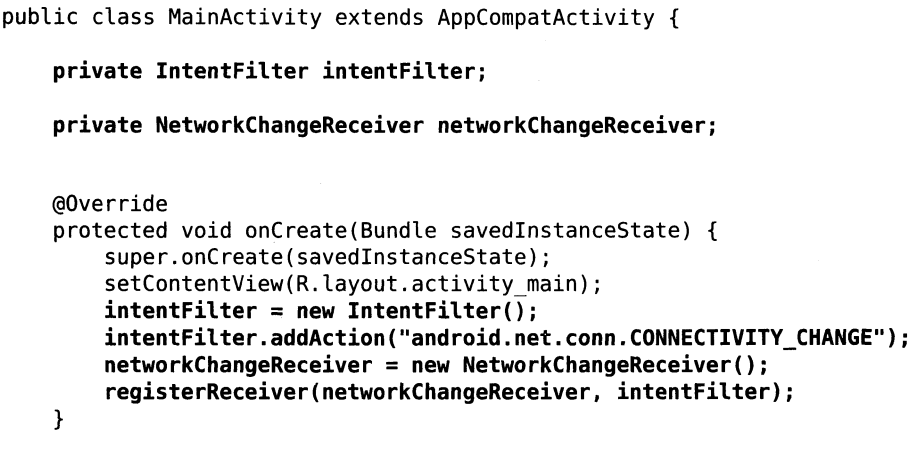
1. **有序广播 在广播发出之后,同一时刻只有一个广播接收器能够收到这条广播消息, 当这个广播接收器中的逻辑执行完毕后,** **广播才会继续传递。有先后顺序，优先级高的先接收且可以截断广播。工作示意图如下：**

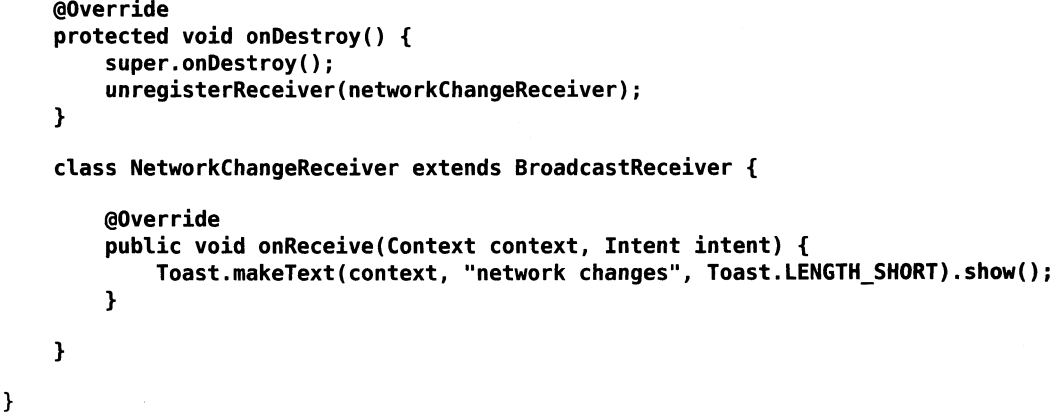


**2 接收系统广播**

1. **动态注册监听网络变化**

**注册广播的方式一般有两种, 在代码中注册和在AndroidManifest.xm1中注册,其中前者也被称为动态注册,后者也被称为静态注册。创建广播接收器的方法是新建一个类, 让它继承自BroadcastReceiver,并重写父类的 onReceive()方法就行了。新建一个 BroadcastTest项日, 然后修改 MainActivity 中的代码, 如下所示:**





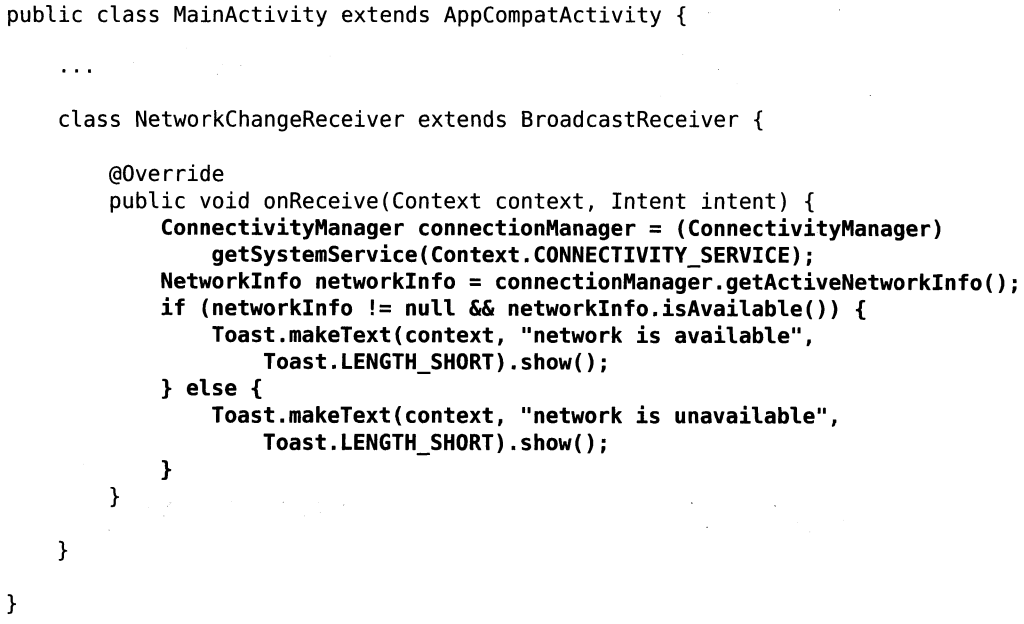
**这里在 MainActivity 中定义了一个内部类 NetworkChangeReceiver,** **每当网络状态发生变化时onReceive( )方法就会得到执行, 这里只是简単地使用 Toast提示了一段文本信息。在onCreate()方法中创建了一个 IntentFilter的实例,并给它添加了一个值为android.net.conn.coNNECTIVITY\_CHANGE的 action,这就是想要监听的广播。接下来创建了一个 NetworkchangeReceiver的实例,然后调用 registerReceiver()方法进行注册,将NetworkChangeReceiver的实例和 IntentFilter的实例都传了进去,这样 NetworkChangeReceiver就会收到所有值为 android.net.conn.CONNECTIVITY\_CHANGE的广播,也就实现了监听网络变化的功能。**

**\*\*\*\*\*\*\*\*\*\*\*\***

**动态注册的广播接收器一定都要取消注册才行。这里是在 onDestroy( )方法中通过调用unregisterReceive r( )方法来实现的。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**还可以对上面的代码进行进一步的优化。修改 MainActivity中的代码,如下所示：**



**在 onReceive( )方法中,首先通过 getSystemService( )方法得到了 connectivityManager的实例,这是一个系统服务类,专门用于管理网络连接的。然后调用它的 getActiveNetworkInfo()方法可以得到 NetworkInfo的实例,接着调用 NetworkInfo的 isAvailable()方法,** **就可以判断出当前是否有网络。**

**\*\*\*\*\*\*\*\*\*\***

**Android系统为了保护用户设备的安全和隐私,做了严格的规定: 如果程序需要进行一些对用户来说比较敏感的操作, 就必须在配置文件中声明权限才可以,否则程序将会直接崩溃。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\***

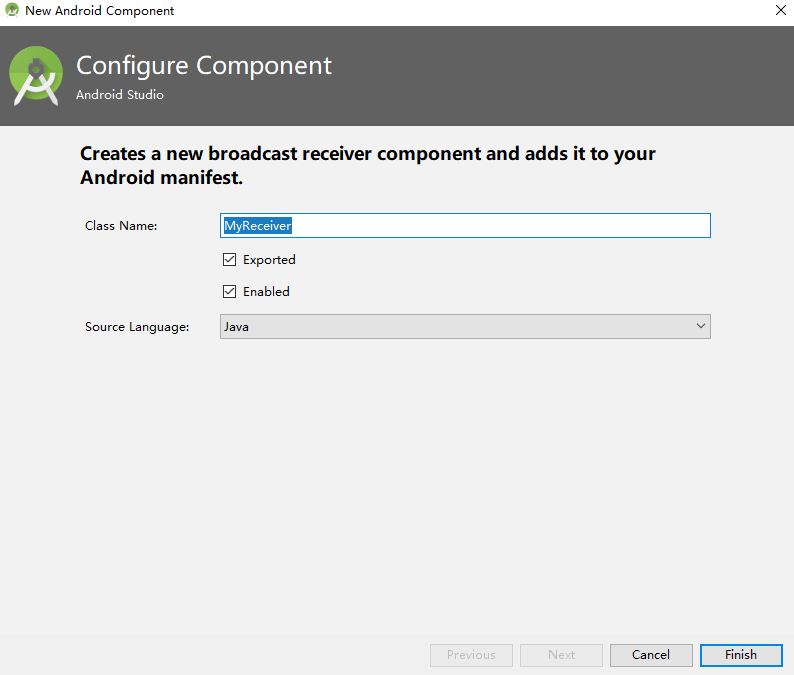
**比如这里访问系统的网络状态就是需要声明权限的。打开AndroidManifest.xm1文件, 在里面加入如下权限就可以访问系统网络状态了。**



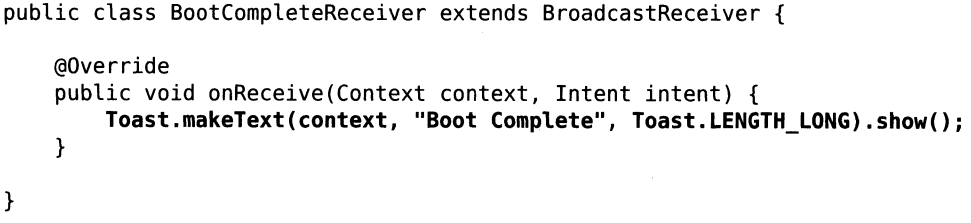


**（2） 静态注册实现开机启动**

**可以使用 Android Studio提供的快捷方式来创建一个广播接收器,右击 com.example.broadcasttest包→New→0ther→Broadcast Receiver,会弹出一个窗口：**

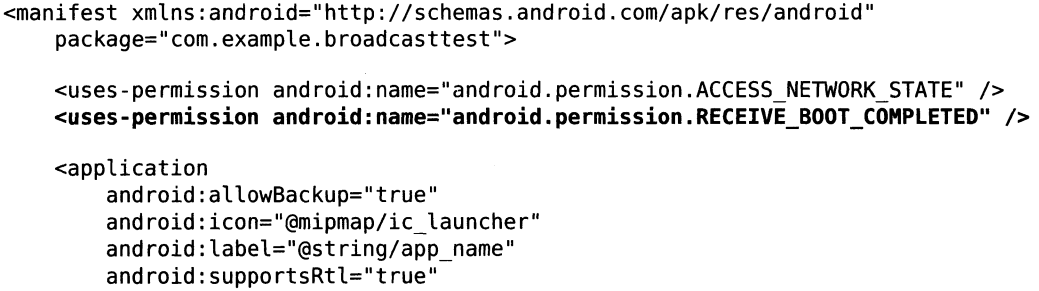


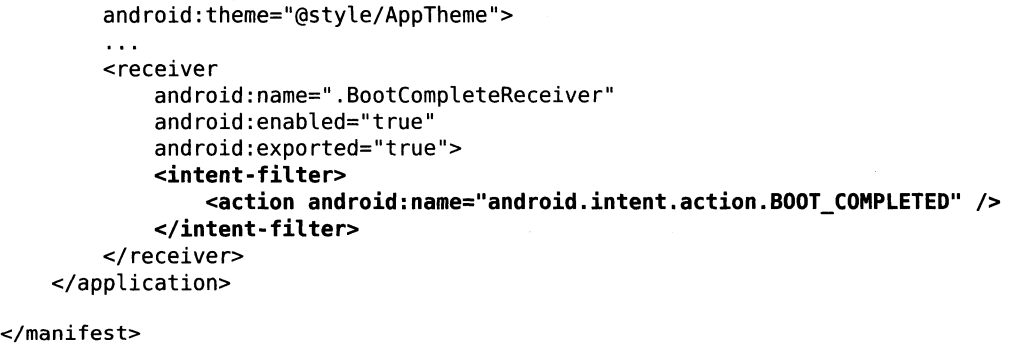
**Exported属性表示是否允许这个广播接收器接收本程序以外的广播, Enabled属性表示是否启用这个广播接收器。勾选这两个属性,点击 Finish完成创建。修改 BootCompleteReceiver中的代码, 如下所示:**



**静态的广播接收器一定要在 AndroidManifest.xm1文件中注册才可以使用,不过由于我是使用Android Studio的快捷方式创建的广播接收器, 因此注册这一步已经被自动完成了。**

**目前BootCompleteReceiver还是不能接收到开机广播的,我们还需要对 AndroidManifest.xm1文件进行修改才行, 如下所示：**





**由于 Android系统启动完成后会发出一条值为 android. intent. action. B00T\_COMPLETED的广播,因此在<intent-fiLter>标签里添加了相应的 action。另外,监听系统开机广播也是需要声明权限的,** **这里使用<uses-perm,ssion>标签又加入了一条 android.** **permission. RECEIVE\_B00T\_COMPLETED权限。**

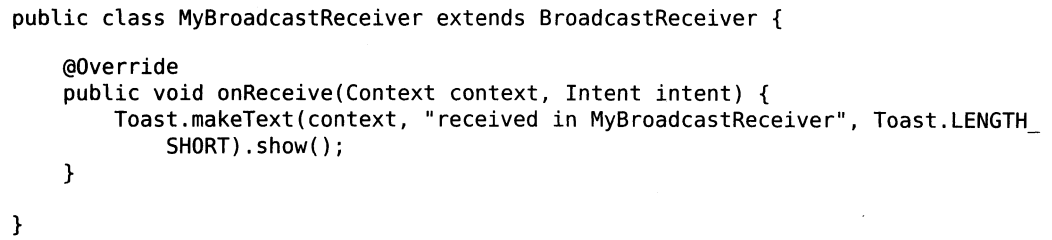
**\*\*\*\*\*\*\*\*\*\*\***

**需要注意的是,** **不要在 onReceive( )方法中添加过多的逻辑或者进行任何的耗时操作, 因为在广播接收器中是不允许开启线程的,当 onReceive()方法运行了较长时间而没有结束时,程序就会报错。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. **发送自定义广播**
2. **发送标准广播**

**先新建一个广播接收器MyBroadcastReceiver,代码如下所示:**

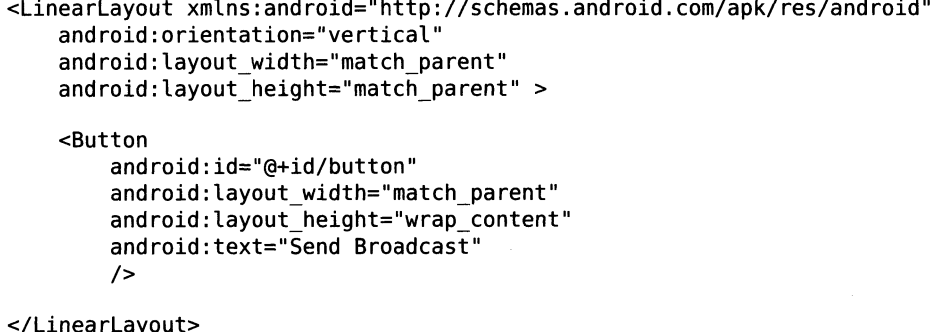


**这里当 MyBroadcastReceiver收到自定义的广播时,就会弹出“received in MyBroadcast\_Receiver”的提示。然后在 AndroidManifest.xml中对这个广播接收器进行修改:**

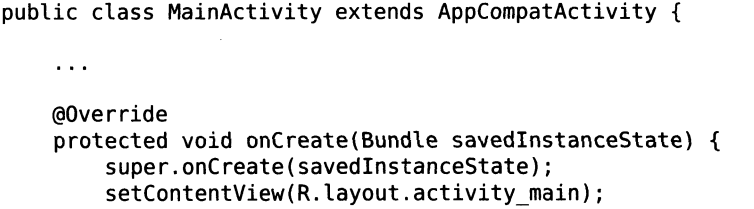
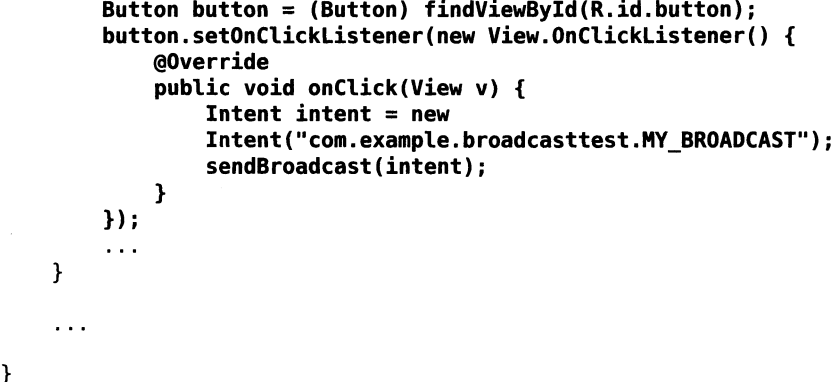


**这里让 MyBroadcastReceiver接收一条值为 com.example.broadcasttest.** **MY BROADCAST的广播。**

**接下来修改 activitly\_main.xm1中的代码,如下所示：**



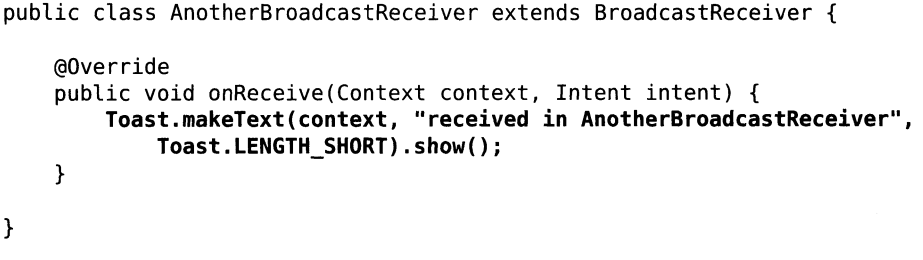
**这里在布局文件中定义了一个按钮, 用于作为发送广播的触发点。然后修改 Mai11lActivity中的代码, 如下所示:**



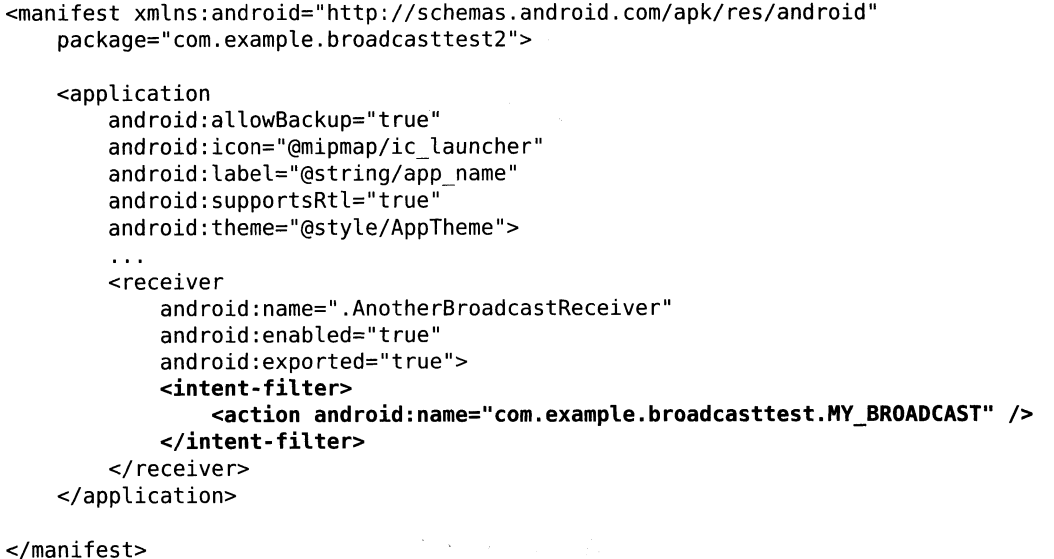
**这里在按钮的点击事件里面加入了发送自定义广播的逻辑。首先构建出了一个Intent对象,并把要发送的广播的值传入,然后调用了 Context的 sendBroadcast()方法将广播发送出去,这样所有监听 com.example. broadcasttest.MY BROADCAST这条广播的广播接收器就会收到消息 。**

1. **发送有序广播**

**广播是一种可以跨进程的通讯方式，在应用程序里发出的广播，其他的应用程序也可以收到。新建一个 BroadcastTest2项目,点击AndroidStudio导航栏→File→New→NewProject进行创建。将项目创建好之后,还需要在这个项目下定义一个广播接收器,用于接收上一小节中的自定义广播。新建AnotherBroadcastReceiver,代码如下所示:**

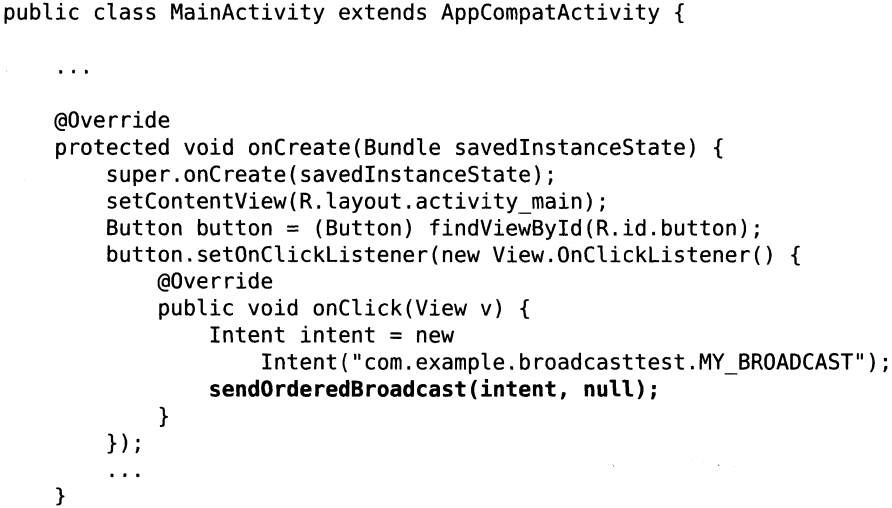


**这里仍然是在广播接收器的 onReceive()方法中弾出了一段文本信息 。然后在AndroidManifest.xm1中对这个广播接收器进行修改,代码如下所示:**



**AnotherBroadcastReceiver同样接收的是 com.example.broadcasttest.MYBROADCAST这条广播。 现在运行 BroadcastTest2项目将这个程序安装到模拟器上, 然后重新回到BroadcastTest项目的主界面,并点击一下 SendBroadcast按钮,就会分别弹出两次提示信息。**

**下面尝试一下发送有序广播。重新回到 BroadcastTest项目,然后修改 MainActivity 中的代码,如下所示：**



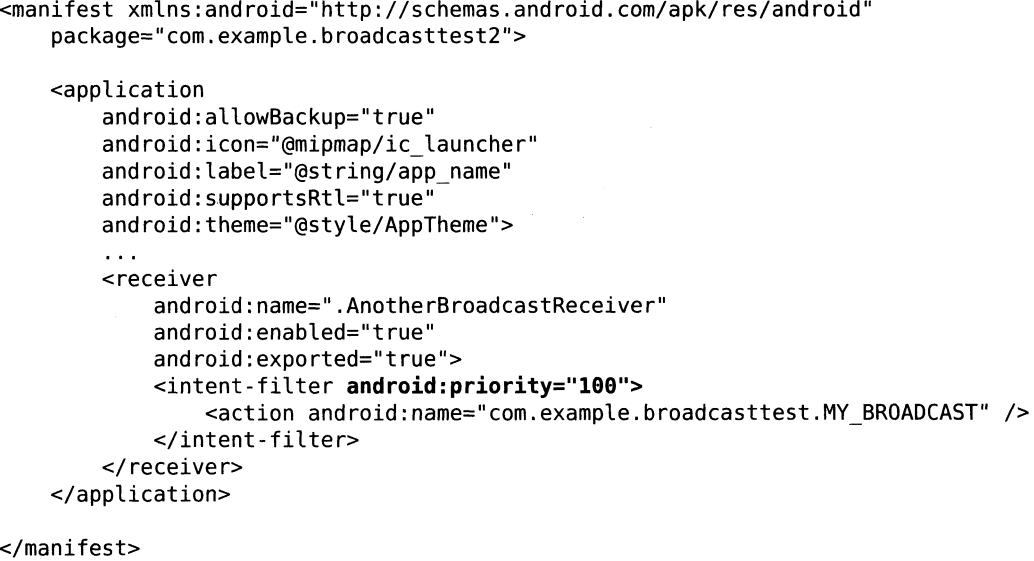
**. . .**

**}**

**这里将sendBroadcast()方法改成 sendOrderedBroadcast( )方法。 sendorderedBroadcast( )方法接收两个参数, 第一个参数仍然是Intent,第二个参数是一个与权限相关的字符串,这里传入 null就行了。**

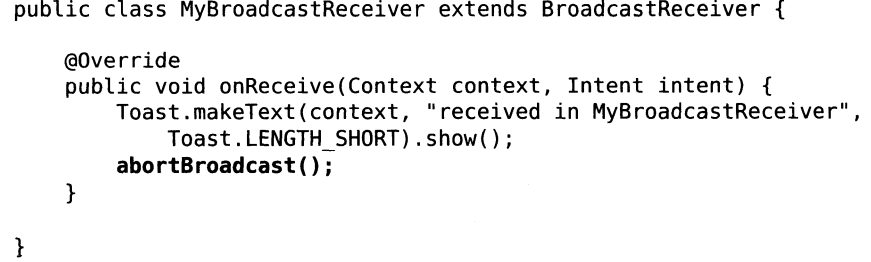
**这个时候的广播接收器是有先后顺序的,** **而且前面的广播接收器还可以将广播截断, 以阻止其继续传播。**

**可以在注册的时候设定广播接收器的顺序，修改AndroidManifest.xm1中的代码,如下所示:**



**通过 and roid: priority属性给广播接收器设置了优先级, 优先级比较高的广播接收器就可以先收到广播 。**

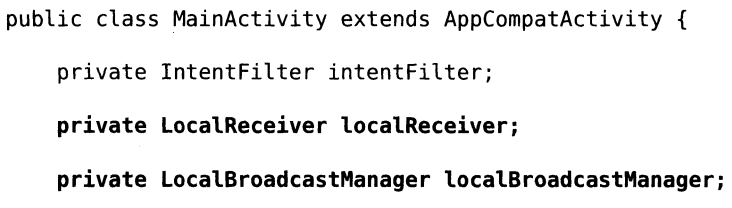
**下面将广播截断，修改MyBroadcastReceiver代码如下：**

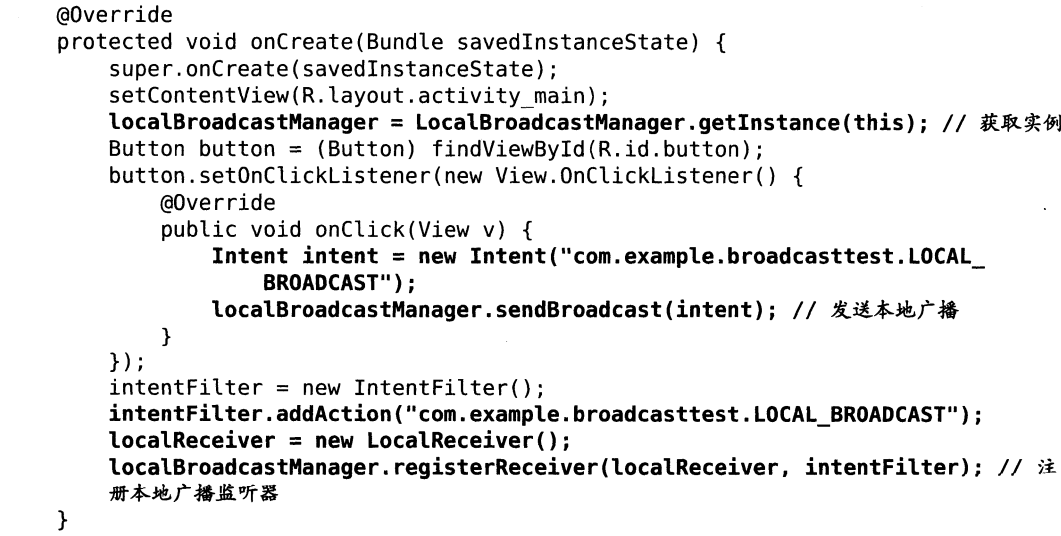


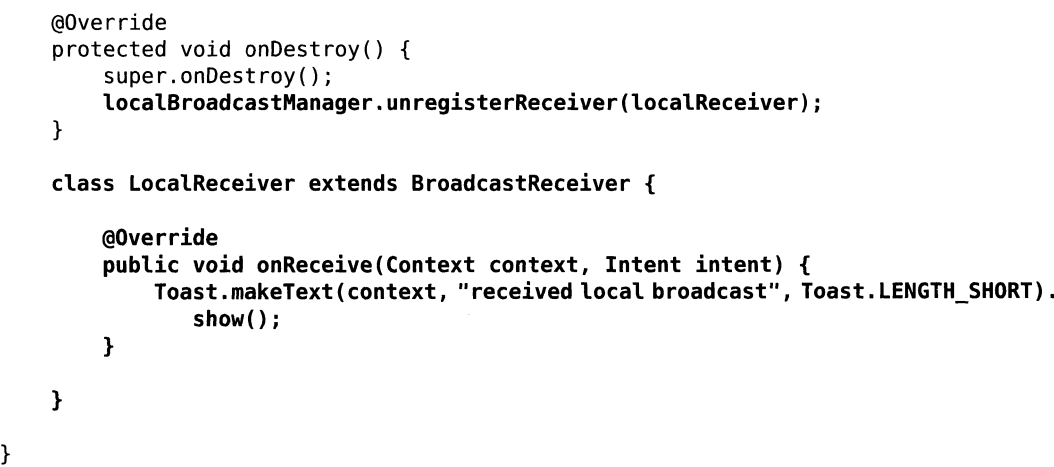
**在 onReceive()方法中调用了 abortBroadcast( )方法,将这条广播截断,后面的广播器无法再接到这条广播。**

1. **使用本地广播**

**了能够简单地解决广播的安全性同题, Android引入了一套本地广播机制, 使用这个机制发出的广播只能够在应用程序的内部进行传递,并且广播接收器也只能接收来自本应用程序发出的广播。它使用了一个 LocalBroadcastManager.来对广播进行管理,** **并提供了发送广播和注册广播接收器的方法。修改 MainActivity 中的代码, 如下所示:**







**首先是通过 LocaIBroadcastManager的 getInstance()方法得到了它的一个实例, 然后在注册广播接收器的时候调用的是 LocalBroadcastManager的registerReceiver()方法,在发送广播的时候调用的是 LocalBroadcastManager的 sendBroadcast()方法。这里在按钮的点击事件里面发出了一条 com.example. broadcasttest.LOCAL\_BROADCAST广播,然后在 LocalReceiver里去接收这条广播。**

**本地广播是无法通过静态注册的方式来接收的，本地广播的优势：**

**\*可以明确地知道正在发送的广播不会离开我们的程序, 因此不必担心机密数据泄漏。**

**\*其他的程序无法将广播发送到我们程序的内部, 因此不需要担心会有安全漏洞的隐患。**

**\*发送本地广播比发送系统全局广播更加高效。**

1. **GIT——初识版本控制工具**

**Git是一个开源的分布式版本控制工具，其开发初衷是为了更好地管理Linux内核。**

1. **安装**
2. **创建代码仓库**

**首先配置一下自己的身份, 这样在提交代码的时候 Git就可以知道是谁提交的了, 命令如下：**

**git config--global user.name"zhulei"**

**git config--global user.email “**[**zl666998@163.com**](mailto:zl666998@163.com)**”**

**然后就可以开始创建代码合库了,仓库( Repository )是用于保存版本管理所需信息的地方, 所有本地提交的代码都会被提交到代码仓库中, 如果有需要还可以再推送到远程仓库中。**

**给 BroadcastBestPractice项日建立一个代码仓库。先进入到 BroadcastBestPractice项目的目录下面,** **然后在这个目录下面输入如下命令:**

**git init**

**仓库创建完成后, 会在 BroadcastBestPractice项目的根目录下生成一个隐藏的.git文件夹,它会记录本地所有的Git操作，可以通过ls-al命令来查看。若要删除本地仓库，只需删除此文件夹即可。**

1. **提交本地代码**

**add用于把想要提交的代码先添加进来**

**git add . 添加项目下的所有文件**

**git add XXXX 添加单个文件或者文件夹**

**commit用于真正的执行提交操作**

**git commit -m “first commit”**

**一定要通过-m参数来加上提交的描述信息,没有描述信息的提交被认为是不合法的。**

1. **小结**

**本章主要是对 Android的广播机制进行了深入的研究,了解了广播的理论知识,** **掌握了接收广播、发送自定义广播以及本地广播的使用方法。广播接收器属于 Android四大组件之一。**