

# Carson带你学Android：BroadcastReceiver史上 最全面解析



Carson\_Ho



关注

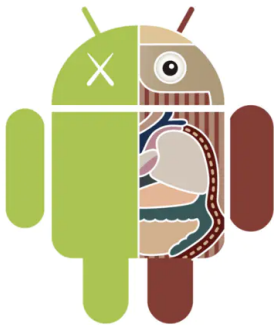


16

2016.10.30 17:43:32

字数 2,607

阅读 149,650

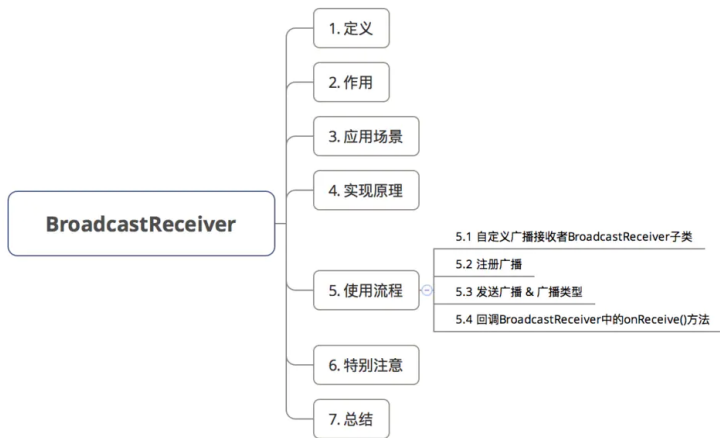


Carson带你学Android

## 前言

- `BroadcastReceiver`（广播接收器），属于 `Android` 四大组件之一
- 在 `Android` 开发中，`BroadcastReceiver` 的应用场景非常多
- 今天，我将详细讲解关于 `BroadcastReceiver` 的一切相关知识

## 目录



示意图

## 1. 定义

即广播，是一个全局的监听器，属于 `Android` 四大组件之一

`Android` 广播分为两个角色：广播发送者、广播接收者

## 2. 作用

监听 / 接收 应用 App 发出的广播消息，并 做出响应

## 3. 应用场景

- Android 不同组件间的通信（含：应用内 / 不同应用之间）
- 多线程通信
- 与 Android 系统在特定情况下的通信

如：电话呼入时、网络可用时

## 4. 实现原理

### 4.1 采用的模型

- Android 中的广播使用了设计模式中的**观察者模式**：基于消息的发布 / 订阅事件模型

因此，Android将广播的**发送者** 和 **接收者** 解耦，使得系统方便集成，更易扩展

### 4.2 模型讲解

- 模型中有3个角色：
  1. 消息订阅者（广播接收者）
  2. 消息发布者（广播发布者）
  3. 消息中心（AMS，即 Activity Manager Service）
- 示意图 & 原理如下



#### 原理描述

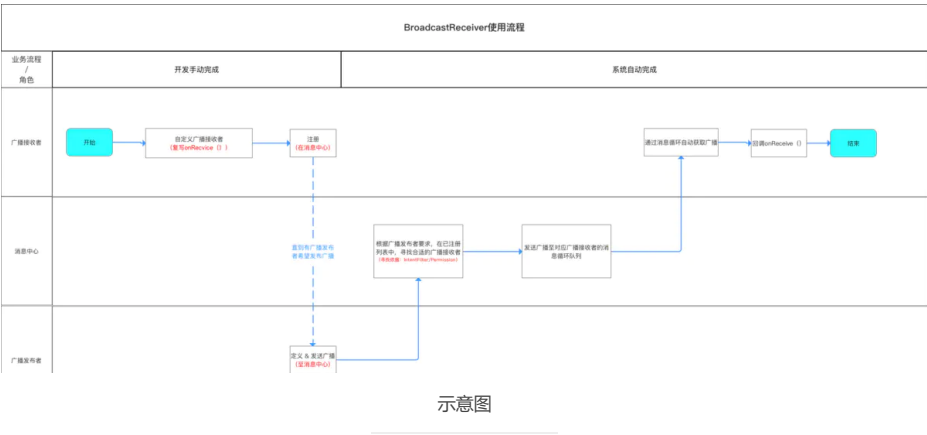
1. 广播接收者 通过 Binder机制在 AMS 注册
2. 广播发送者 通过 Binder 机制向 AMS 发送广播
3. AMS 根据 广播发送者 要求，在已注册列表中，寻找合适的广播接收者  
(寻找依据: IntentFilter / Permission)
4. AMS将广播发送到合适的广播接收者相应的消息循环队列中
5. 广播接收者通过 消息循环 拿到此广播，并回调 onReceive()

特别注意：广播发送者 和 广播接收者的执行 是 \*异步的，即 广播发送者 不会关心有无接收者接收 & 也不确定接收者何时才能接收到

示意图

## 5. 使用流程

- 使用流程如下：



- 下面，我将一步步介绍如何使用 `BroadcastReceiver`

即上图中的 **开发者手动完成部分**

### 5.1 自定义广播接收者BroadcastReceiver

- 继承 `BroadcastReceiver` 基类
- 必须复写抽象方法 `onReceive()` 方法

- 广播接收器接收到相应广播后，会自动回调 `onReceive()` 方法
- 一般情况下，`onReceive` 方法会涉及 与 其他组件之间的交互，如发送 `Notification`、启动 `Service` 等
- 默认情况下，广播接收器运行在 `UI` 线程，因此，`onReceive()` 方法不能执行耗时操作，否则将导致 `ANR`

- 代码范例

mBroadcastReceiver.java

```
1 // 继承BroadcastReceiver基类
2 public class mBroadcastReceiver extends BroadcastReceiver {
3
4     // 复写onReceive()方法
5     // 接收到广播后，则自动调用该方法
6     @Override
7     public void onReceive(Context context, Intent intent) {
8         // 写入接收广播后的操作
9     }
10 }
11
```

### 5.2 广播接收器注册

注册的方式分为两种：静态注册、动态注册

#### 5.2.1 静态注册

- 注册方式：在AndroidManifest.xml里通过<receive>标签声明
- 属性说明：

```
1 <receiver
2     android:enabled=["true" | "false"]
3 //此broadcastReceiver能否接收其他App的发出的广播
```

```

4  //默认值是由receiver中有无intent-filter决定的: 如果有intent-filter, 默认值为true, 否则为false
5      android:exported=["true" | "false"]
6      android:icon="drawable resource"
7      android:label="string resource"
8  //继承BroadcastReceiver子类的类名
9      android:name=".mBroadcastReceiver"
10 //具有相应权限的广播发送者发送的广播才能被此BroadcastReceiver所接收:
11      android:permission="string"
12 //BroadcastReceiver运行所处的进程
13 //默认为app的进程, 可以指定独立的进程
14 //注: Android四大基本组件都可以通过此属性指定自己的独立进程
15      android:process="string" >
16
17 //用于指定此广播接收器将接收的广播类型
18 //本示例中给出的是用于接收网络状态改变时发出的广播
19 <intent-filter>
20 <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
21 </intent-filter>
22 </receiver>

```

## • 注册示例

```

1  <receiver
2      //此广播接收者类是mBroadcastReceiver
3      android:name=".mBroadcastReceiver" >
4      //用于接收网络状态改变时发出的广播
5      <intent-filter>
6          <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
7      </intent-filter>
8  </receiver>

```

当此 App 首次启动时, 系统会自动实例化 `mBroadcastReceiver` 类, 并注册到系统中。

## 5.2.2 动态注册

- 注册方式: 在代码中调用 `Context.registerReceiver()` 方法
- 具体代码如下:

```

1  // 选择在Activity生命周期方法中的onResume()中注册
2  @Override
3      protected void onResume(){
4          super.onResume();
5
6          // 1. 实例化BroadcastReceiver子类 & IntentFilter
7          mBroadcastReceiver mBroadcastReceiver = new mBroadcastReceiver();
8          IntentFilter intentFilter = new IntentFilter();
9
10         // 2. 设置接收广播的类型
11         intentFilter.addAction(android.net.conn.CONNECTIVITY_CHANGE);
12
13         // 3. 动态注册: 调用Context的registerReceiver()方法
14         registerReceiver(mBroadcastReceiver, intentFilter);
15     }
16
17
18     // 注册广播后, 要在相应位置记得销毁广播
19     // 即在onPause() 中unregisterReceiver(mBroadcastReceiver)
20     // 当此Activity实例化时, 会动态将MyBroadcastReceiver注册到系统中
21     // 当此Activity销毁时, 动态注册的MyBroadcastReceiver将不再接收到相应的广播。
22     @Override
23     protected void onPause() {
24         super.onPause();
25         // 销毁在onResume()方法中的广播
26         unregisterReceiver(mBroadcastReceiver);
27     }
28 }

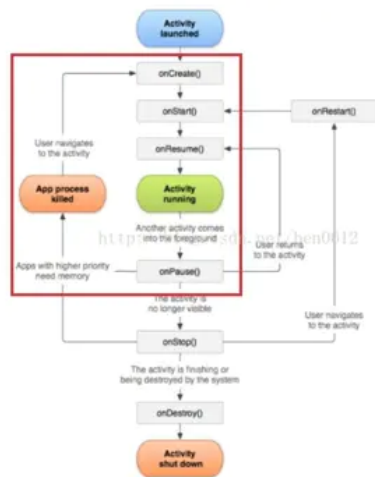
```

## 特别注意

- 动态广播最好在 Activity 的 onResume() 注册、 onPause() 注销。
- 原因：
  1. 对于动态广播，有注册就必然得有注销，否则会导致内存泄露

重复注册、重复注销也不允许

2. Activity 生命周期如下：



Activity生命周期

Activity生命周期的方法是成对出现的：

- onCreate() & onDestroy()
- onStart() & onStop()
- onResume() & onPause()

在onResume()注册、 onPause()注销是因为onPause()在App死亡前一定会被执行，从而保证广播在App死亡前一定会被注销，从而防止内存泄露。

1. 不在onCreate() & onDestroy() 或 onStart() & onStop()注册、注销是因为：  
当系统因为内存不足（优先级更高的应用需要内存，请看上图红框）要回收Activity占用的资源时，Activity在执行完onPause()方法后就会被销毁，有些生命周期方法onStop(), onDestroy()就不会执行。当再回到此Activity时，是从onCreate方法开始执行。
2. 假设我们将广播的注销放在onStop(), onDestroy()方法里的话，有可能在Activity被销毁后还未执行onStop(), onDestroy()方法，即广播仍还未注销，从而导致内存泄露。
3. 但是， onPause()一定会被执行，从而保证了广播在App死亡前一定会被注销，从而防止内存泄露。

5.2.3 两种注册方式的区别

注册方式	区别		
	使用方式	特点	应用场景
静态注册 (常驻广播)	在AndroidManifest.xml里通过<receive>标签声明	<ul style="list-style-type: none"><li>• 常驻，不受任何组件的生命周期影响</li><li>(应用程序关闭后，如果有信息广播来，程序依旧会被系统调用)</li><li>• 缺点：耗电、占内存</li></ul>	需要时刻监听广播
动态注册 (非常驻广播)	在代码中调用Context.registerReceiver () 方法	<ul style="list-style-type: none"><li>• 非常驻，灵活，跟随组件的生命周期变化</li><li>(组件结束=&gt;广播结束，在组件结束前，必须移除广播接收器)</li></ul>	需要特定时刻监听广播

## 5.3 广播发送者向AMS发送广播

### 5.3.1 广播的发送

- 广播是用“意图（`Intent`）”标识
- 定义广播的本质 = 定义广播所具备的“意图（`Intent`）”
- 广播发送 = 广播发送者 将此广播的“意图（`Intent`）”通过`sendBroadcast()`方法发送出去

### 5.3.2 广播的类型

广播的类型主要分为5类：

- 普通广播（`Normal Broadcast`）
- 系统广播（`System Broadcast`）
- 有序广播（`Ordered Broadcast`）
- 粘性广播（`Sticky Broadcast`）
- App应用内广播（`Local Broadcast`）

具体说明如下：

#### 1. 普通广播（Normal Broadcast）

即 开发者自身定义 `intent` 的广播（最常用）。发送广播使用如下：

```
1 Intent intent = new Intent();
2 //对应BroadcastReceiver中IntentFilter的action
3 intent.setAction(BROADCAST_ACTION);
4 //发送广播
5 sendBroadcast(intent);
```

- 若被注册了的广播接收者中注册时 `intentFilter` 的 `action` 与上述匹配，则会接收此广播（即进行回调 `onReceive()`）。如下 `mBroadcastReceiver` 则会接收上述广播

```
1 <receiver
2     //此广播接收者类是mBroadcastReceiver
3     android:name=".mBroadcastReceiver" >
4     //用于接收网络状态改变时发出的广播
5     <intent-filter>
6         <action android:name="BROADCAST_ACTION" />
7     </intent-filter>
8 </receiver>
```

- 若发送广播有相应权限，那么广播接收者也需要相应权限

#### 2. 系统广播（System Broadcast）

- Android中内置了多个系统广播：只要涉及到手机的基本操作（如开机、网络状态变化、拍照等等），都会发出相应的广播
- 每个广播都有特定的Intent - Filter（包括具体的action），Android系统广播action如下：

系统操作	action
监听网络变化	android.net.conn.CONNECTIVITY_CHANGE
关闭或打开飞行模式	Intent.ACTION_AIRPLANE_MODE_CHANGED

系统操作	action
充电时或电量发生变化	Intent.ACTION_BATTERY_CHANGED
电池电量低	Intent.ACTION_BATTERY_LOW
电池电量充足（即从电量低变化到饱满时会发出广播	Intent.ACTION_BATTERY_OKAY
系统启动完成后(仅广播一次)	Intent.ACTION_BOOT_COMPLETED
按下照相时的拍照按键(硬件按键)时	Intent.ACTION_CAMERA_BUTTON
屏幕锁屏	Intent.ACTION_CLOSE_SYSTEM_DIALOGS
设备当前设置被改变时(界面语言、设备方向等)	Intent.ACTION_CONFIGURATION_CHANGED
插入耳机时	Intent.ACTION_HEADSET_PLUG
未正确移除SD卡但已取出来时(正确移除方法:设置--SD卡和设备内存--卸载SD卡)	Intent.ACTION_MEDIA_BAD_REMOVAL
插入外部储存装置（如SD卡）	Intent.ACTION_MEDIA_CHECKING
成功安装APK	Intent.ACTION_PACKAGE_ADDED
成功删除APK	Intent.ACTION_PACKAGE_REMOVED
重启设备	Intent.ACTION_REBOOT
屏幕被关闭	Intent.ACTION_SCREEN_OFF
屏幕被打开	Intent.ACTION_SCREEN_ON
关闭系统时	Intent.ACTION_SHUTDOWN
重启设备	Intent.ACTION_REBOOT

注：当使用系统广播时，只需要在注册广播接收者时定义相关的action即可，并不需要手动发送广播，当系统有相关操作时会自动进行系统广播

3. 有序广播（Ordered Broadcast）

- 定义  
发送出去的广播被广播接收者按照先后顺序接收

有序是针对广播接收者而言的

- 广播接受者接收广播的顺序规则（同时面向静态和动态注册的广播接受者）
  1. 按照Priority属性值从大-小排序；
  2. Priority属性相同者，动态注册的广播优先；
- 特点
  1. 接收广播按顺序接收
  2. 先接收的广播接收者可以对广播进行截断，即后接收的广播接收者不再接收到此广播；
  3. 先接收的广播接收者可以对广播进行修改，那么后接收的广播接收者将接收到被修改后的广播

- 具体使用

有序广播的使用过程与普通广播非常类似，差异仅在于广播的发送方式：

```
1 | sendOrderedBroadcast(intent);
```

#### 4. App应用内广播 (Local Broadcast)

- 背景

Android中的广播可以跨App直接通信（exported对于有intent-filter情况下默认值为true）

- 冲突

可能出现的问题：

- 其他App针对性发出与当前App intent-filter相匹配的广播，由此导致当前App不断接收广播并处理；
- 其他App注册与当前App一致的intent-filter用于接收广播，获取广播具体信息；即会出现安全性 & 效率性的问题。

- 解决方案

使用App应用内广播 (Local Broadcast)

1. App应用内广播可理解作为一种局部广播，广播的发送者和接收者都同属于一个App。
2. 相比于全局广播（普通广播），App应用内广播优势体现在：安全性高 & 效率高

- 具体使用1 - 将全局广播设置成局部广播

1. 注册广播时将exported属性设置为false，使得非本App内部发出的此广播不被接收；
2. 在广播发送和接收时，增设相应权限permission，用于权限验证；
3. 发送广播时指定该广播接收器所在的包名，此广播将只会发送到此包中的App内与之相匹配的有效广播接收器中。

通过intent.setPackage(packageName)指定报名

- 具体使用2 - 使用封装好的LocalBroadcastManager类

使用方式上与全局广播几乎相同，只是注册/取消注册广播接收器和发送广播时将参数的context变成了LocalBroadcastManager的单一实例

注：对于LocalBroadcastManager方式发送的应用内广播，只能通过LocalBroadcastManager动态注册，不能静态注册

```
1 | //注册应用内广播接收器
2 | //步骤1: 实例化BroadcastReceiver子类 & IntentFilter mBroadcastReceiver
3 | mBroadcastReceiver = new mBroadcastReceiver();
4 | IntentFilter intentFilter = new IntentFilter();
5 |
6 | //步骤2: 实例化LocalBroadcastManager的实例
7 | localBroadcastManager = LocalBroadcastManager.getInstance(this);
8 |
9 | //步骤3: 设置接收广播的类型
10 | intentFilter.addAction(android.net.conn.CONNECTIVITY_CHANGE);
11 |
12 | //步骤4: 调用LocalBroadcastManager单一实例的registerReceiver () 方法进行动态注册
13 | localBroadcastManager.registerReceiver(mBroadcastReceiver, intentFilter);
14 |
```



```
15 //取消注册应用内广播接收器
16 localBroadcastManager.unregisterReceiver(mBroadcastReceiver);
17
18 //发送应用内广播
19 Intent intent = new Intent();
20 intent.setAction(BROADCAST_ACTION);
21 localBroadcastManager.sendBroadcast(intent);
22
```

## 5. 粘性广播 (Sticky Broadcast)

由于在Android5.0 & API 21中已经失效，所以不建议使用，在这里也不作过多的总结。

## 6. 特别注意

对于不同注册方式的广播接收器回调onReceive (Context context, Intent intent) 中的context返回值是不一样的：

- 对于静态注册（全局+应用内广播），回调onReceive(context, intent)中的context返回值是：ReceiverRestrictedContext；
- 对于全局广播的动态注册，回调onReceive(context, intent)中的context返回值是：Activity Context；
- 对于应用内广播的动态注册（LocalBroadcastManager方式），回调onReceive(context, intent)中的context返回值是：Application Context。
- 对于应用内广播的动态注册（非LocalBroadcastManager方式），回调onReceive(context, intent)中的context返回值是：Activity Context；