4.3.2 BroadcastReceiver庖丁解牛

分类 **Android 基础入门教程**

本节引言：

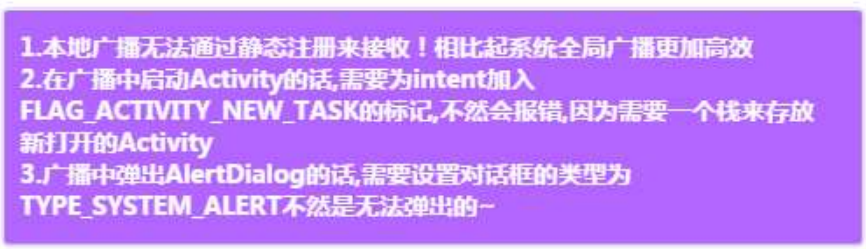
上节我们对BroadcastReceiver已经有了一个初步的了解了，知道两种广播类型：标准与有序， 动态或静态注册广播接收者，监听系统广播，自己发送广播！已经满足我们的基本需求了~ 但是前面写的广播都是全局广播！这同样意味着我们APP发出的广播，其他APP都会接收到， 或者其他APP发送的广播，我们的APP也同样会接收到，这样容易引起一些安全性的问题！而 Android中给我们提供了本地广播的机制，使用该机制发出的广播只会在APP内部传播，而且 广播接收者也只能收到本应用发出的广播！

1.本地广播

1）核心用法：

PS：本地广播无法通过静态注册方式来接受,相比起系统全局广播更加高效

2）注意事项：



3）代码示例(别处登陆踢用户下线)：

像微信一样，正在运行的微信，如果我们用别的手机再次登陆自己的

账号，前面这个是会提醒账户 在别的终端登录这样，然后把我们打开的所有Activity都关掉，然后回到登陆页面这样~
下面我们就来写个简单的例子：

运行效果图：

代码实现：

Step 1：准备一个关闭所有Activity的ActivityCollector ，这里之前用前面Activity提供的那个!

ActivityCollector.java

```
public class ActivityCollector {
    private static List<Activity> activities = new ArrayList<Activity>();
    public static void addActivity(Activity activity) {
        activities.add(activity);
    }
    public static void removeActivity(Activity activity) {
        activities.remove(activity);
    }
    public static void finishAll() {
        for (Activity activity : activities) {
            if (!activity.isFinishing()) {
                activity.finish();
            }
        }
    }
}
```

Step 2：先写要给简单的BaseActivity，用来继承，接着写下登陆界面！

```
public class BaseActivity extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityCollector.addActivity(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        ActivityCollector.removeActivity(this);
    }
}
```

LoginActivity.java:

```
public class LoginActivity extends BaseActivity implements View
.OnClickListener{


    private SharedPreferences pref;
    private SharedPreferences.Editor editor;

    private EditText edit_user;
    private EditText edit_pawd;
    private Button btn_login;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        pref = PreferenceManager.getDefaultSharedPreferences(th
is);

        bindViews();
    }

    private void bindViews() {
        edit_user = (EditText) findViewById(R.id.edit_user);
        edit_pawd = (EditText) findViewById(R.id.edit_pawd);
        btn_login = (Button) findViewById(R.id.btn_login);
        btn_login.setOnClickListener(this);
    }

    @Override
    protected void onStart() {
        super.onStart();
        if(!pref.getString("user","").equals("")){
            edit_user.setText(pref.getString("user",""));
            edit_pawd.setText(pref.getString("pawd",""));
        }
    }
```

```
    @Override
    public void onClick(View v) {
        String user = edit_user.getText().toString();
        String pawd = edit_pawd.getText().toString();
        if(user.equals("123")&&pawd.equals("123")){
            editor = pref.edit();
            editor.putString("user", user);
            editor.putString("pawd", pawd);
            editor.commit();
            Intent intent = new Intent(LoginActivity.this, Main
Activity.class);
            startActivity(intent);
            Toast.makeText(LoginActivity.this,"哟，竟然蒙对了~",T
oast.LENGTH_SHORT).show();
            finish();
        }else{
            Toast.makeText(LoginActivity.this,"这么简单都输出，脑
子呢？",Toast.LENGTH_SHORT).show();
        }

    }
}
```

Step 3：自定义一个BroadcastReceiver，在onReceive里完成弹出对话框操作，以及启动登陆页面： **MyBcReceiver.java**

```
public class MyBcReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(final Context context, Intent intent)
 {
        AlertDialog.Builder dialogBuilder = new AlertDialog.Bui
lder(context);
        dialogBuilder.setTitle("警告: ");
        dialogBuilder.setMessage("您的账号在别处登录，请重新登陆~")
;
        dialogBuilder.setCancelable(false);
        dialogBuilder.setPositiveButton("确定",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
 int which) {
                        ActivityCollector.finishAll();
                        Intent intent = new Intent(context, Log
inActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_NE
W_TASK);
                        context.startActivity(intent);
                    }
                });
        AlertDialog alertDialog = dialogBuilder.create();
        alertDialog.getWindow().setType(
                WindowManager.LayoutParams.TYPE_SYSTEM_ALERT);
        alertDialog.show();
```

```
    }
}
```

别忘了AndroidManifest.xml中加上系统对话框权限： **< uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />**

Step 4：在MainActivity中，实例化localBroadcastManager，拿他完成相关操作，另外销毁时 注意unregisterReceiver！

MainActivity.java

```
public class MainActivity extends BaseActivity {

    private MyBcReceiver localReceiver;
    private LocalBroadcastManager localBroadcastManager;
    private IntentFilter intentFilter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        localBroadcastManager = LocalBroadcastManager.getInstan
ce(this);

        //初始化广播接收者，设置过滤器
        localReceiver = new MyBcReceiver();
        intentFilter = new IntentFilter();
        intentFilter.addAction("com.jay.mybcreceiver.LOGIN_OTHE
R");
        localBroadcastManager.registerReceiver(localReceiver, i
ntentFilter);

        Button btn_send = (Button) findViewById(R.id.btn_send);
        btn_send.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                Intent intent = new Intent("com.jay.mybcreceive
r.LOGIN_OTHER");
                localBroadcastManager.sendBroadcast(intent);
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        localBroadcastManager.unregisterReceiver(localReceiver)
;
    }
}
```

好的，就是这么简单，别忘记注册Activity哦~

2.Android 4.3以上版本监听开机启动广播的问题解决：

在Android 4.3以上的版本，允许我们将应用安装在SD上，我们都知道是系统开机 间隔一小段时间后，才装载SD卡的，这样我们的应用就可能监听不到这个广播了！ 所以我们需要既监听开机广播又监听SD卡挂载广播！
另外，有些手机可能并没有SD卡，所以这两个广播监听我们不能写到同一个Intetn-filter里 而是应该写成两个，配置代码如下：

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLE
TED"/>
    </intent-filter>
    <intent-filter>
        <action android:name="ANDROID.INTENT.ACTION.MEDIA_MOUNT
ED"/>
        <action android:name="ANDROID.INTENT.ACTION.MEDIA_UNMOU
NTED"/>
        <data android:scheme="file"/>
    </intent-filter>
</receiver>
```

3.常用的系统广播总结：

最后给大家提供下我们平常可能会用到的一些系统广播吧：

```
Intent.ACTION_AIRPLANE_MODE_CHANGED;
//关闭或打开飞行模式时的广播

<strong>Intent.ACTION_BATTERY_CHANGED;
//充电状态，或者电池的电量发生变化
//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Co
ntext.registerReceiver()注册

<strong>Intent.ACTION_BATTERY_LOW;
//表示电池电量低

<strong>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

Intent.ACTION_BOOT_COMPLETED;
//在系统启动完成后，这个动作被广播一次（只有一次）。

Intent.ACTION_CAMERA_BUTTON;
//按下照相时的拍照按键(硬件按键)时发出的广播

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

```
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)

Intent.ACTION_DATE_CHANGED;
//设备日期发生改变时会发出此广播

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？

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？

Intent.ACTION_DOCK_EVENT;
//
//发出此广播的地方frameworks\base\services\java\com\android\server\DockObserver.java

Intent.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE;
////移动APP完成之后，发出的广播(移动是指:APP2SD)

Intent.ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE;
//正在移动APP时，发出的广播(移动是指:APP2SD)

Intent.ACTION_GTALK_SERVICE_CONNECTED;
//Gtalk已建立连接时发出的广播

Intent.ACTION_GTALK_SERVICE_DISCONNECTED;
//Gtalk已断开连接时发出的广播

Intent.ACTION_HEADSET_PLUG;
//在耳机口上插入耳机时发出的广播

Intent.ACTION_INPUT_METHOD_CHANGED;
//改变输入法时发出的广播

Intent.ACTION_LOCALE_CHANGED;
//设备当前区域设置已更改时发出的广播

Intent.ACTION_MANAGE_PACKAGE_STORAGE;
//

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
//广播：扩展介质（扩展卡）已经从 SD 卡插槽拔出，但是挂载点 (mount point) 还没解除 (unmount)

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)
```

```
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?
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?
//广播：用户想要移除扩展介质（拔掉扩展卡）。
Intent.ACTION_MEDIA_MOUNTED;
//插入SD卡并且已正确安装（识别）时发出的广播
//广播：扩展介质被插入，而且已经被挂载。
Intent.ACTION_MEDIA_NOFS;
//
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?
// 广播：扩展介质被移除。
Intent.ACTION_MEDIA_SCANNER_FINISHED;
//广播：已经扫描完介质的一个目录
Intent.ACTION_MEDIA_SCANNER_SCAN_FILE;
//
Intent.ACTION_MEDIA_SCANNER_STARTED;
//广播：开始扫描介质的一个目录

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。
 Intent.ACTION_MEDIA_UNMOUNTABLE;
//
Intent.ACTION_MEDIA_UNMOUNTED
// 广播：扩展介质存在，但是还没有被挂载 (mount)。
Intent.ACTION_NEW_OUTGOING_CALL;

Intent.ACTION_PACKAGE_ADDED;
//成功的安装APK之后
//广播：设备上新安装了一个应用程序包。
//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）
 Intent.ACTION_PACKAGE_CHANGED;
//一个已存在的应用程序包已经改变，包括包名
Intent.ACTION_PACKAGE_DATA_CLEARED;
//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?)
//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）

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?
//
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）

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？）
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）
Intent.ACTION_POWER_CONNECTED;
//插上外部电源时发出的广播
Intent.ACTION_POWER_DISCONNECTED;
//已断开外部电源连接时发出的广播
Intent.ACTION_PROVIDER_CHANGED;
//

Intent.ACTION_REBOOT;
//重启设备时的广播

Intent.ACTION_SCREEN_OFF;
//屏幕被关闭之后的广播

Intent.ACTION_SCREEN_ON;
//屏幕被打开之后的广播

Intent.ACTION_SHUTDOWN;
//关闭系统时发出的广播

Intent.ACTION_TIMEZONE_CHANGED;
//时区发生改变时发出的广播

Intent.ACTION_TIME_CHANGED;
//时间被设置时发出的广播

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

Intent.ACTION_UID_REMOVED;
//一个用户ID已经从系统中移除发出的广播
//

Intent.ACTION_UMS_CONNECTED;
//设备已进入USB大容量储存状态时发出的广播？

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？

Intent.ACTION_USER_PRESENT;
//

Intent.ACTION_WALLPAPER_CHANGED;
//设备墙纸已改变时发出的广播
```

4.本节小结：

好的，关于BroadcastReceiver的学习就到这里，如果你有什么补充或者建议，欢迎提出~

万分感激~

← 4.3.1 BroadcastReceiver牛刀小试 4.4.1 ContentProvider初探 →

在线实例
· HTML 实例
· CSS 实例
· JavaScript 实例
· Ajax 实例
· jQuery 实例
· XML 实例
· Java 实例

字符集&工具
· HTML 字符集设置
· HTML ASCII 字符集
· HTML ISO-8859-1
· HTML 实体符号
· HTML 拾色器
· JSON 格式化工具

最新更新
· Swift 正式开源
· PHP 7 正式发布
· Shell 编程快速入门
· Shell 文件包含
· Shell 输入/输出...
· Shell printf 命令
· Shell 基本运算符

站点信息
· 意见反馈
· 免责声明
· 关于我们
· 文章归档

关注微信