< 2017年10月 >

日	一	二	三	四	五	六
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Android中Service的使用详解和注意点（LocalService）
Posted on 2013-09-02 13:27 work hard work smart 阅读(47253) 评论(8) 编辑 收藏

Android中Service的使用详解和注意点（LocalService）

原文地址

开始，先稍稍讲一点android中Service的概念和用途吧~

Service分为本地服务（LocalService）和远程服务（RemoteService）：

1、本地服务依附在主进程上而不是独立的进程，这样在一定程度上节约了资源，另外Local服务因为是在同一进程因此不需要IPC，

也不需要AIDL。相应bindService会方便很多。主进程被Kill后，服务便会终止。

2、远程服务为独立的进程，对应进程名格式为所在包名加上你指定的android:process字符串。由于是独立的进程，因此在Activity所在进程被Kill的时候，该服务依然在运行，

不受其他进程影响，有利于为多个进程提供服务具有较高的灵活性。该服务是独立的进程，会占用一定资源，并且使用AIDL进行IPC稍微麻烦一点。

按使用方式可以分为以下三种：

1、startService 启动的服务：主要用于启动一个服务执行后台任务，不进行通信。停止服务使用stopService；

2、bindService 启动的服务：该方法启动的服务可以进行通信。停止服务使用unbindService；

3、startService 同时也 bindService 启动的服务：停止服务应同时使用stepService与unbindService

Service 与 Thread 的区别

很多时候，你可能会问，为什么要用 Service，而不用 Thread 呢，因为用 Thread 是很方便的，比起 Service 也方便多了，下面我详细的来解释一下。

1). Thread：Thread 是程序执行的最小单元，它是分配CPU的基本单位。可以用 Thread 来执行一些异步的操作。

2). Service：Service 是android的一种机制，当它运行的时候如果是Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上。因此请不要把 Service 理解成线程，它跟线程半毛钱的关系都没有！

更多链接

我的标签

Android (2)
Android 基础 (2)
c#基础知识----委托和事件 (2)
Design Model (1)
STL (1)
Word 2007 基本操作 (1)
数据结构和算法 (1)
ATL (1)
c# (1)

随笔分类(534)

.NET互操作性(8)
A.设计模式(1)
B.WPF(32)
B.WPF 控件开发(10)
B.WPF 类库整理
C.Blend(WPF/Silverlight)(1)
D.C# (36)
D.Effective C#(1)
D.More Effective C#
E.ASP.NET开发
E.微信公众号开发（ASP.NET ）(2)
EPub(5)
F.VS 工具使用技巧(11)
G.Experience Sharing(8)
H.ATL
I.COM(1)
J.CPP(31)
J.CPP 高级(3)
K.WCF(1)
L.WF
M.数据结构(1)
N.work hard work smart 随笔目录(1)
O. HTML5(7)
O.CSS3(3)
O.JQuery(8)
P. Win 8(34)
Q.JavaScript(4)

既然这样，那么我们为什么要用 Service 呢？其实这跟 android 的系统机制有关，我们先拿 Thread 来说。Thread 的运行是独立于 Activity 的，也就是说当一个 Activity 被 finish 之后，如果你没有主动停止 Thread 或者 Thread 里的 run 方法没有执行完毕的话，Thread 也会一直执行。因此这里会出现一个问题：当 Activity 被 finish 之后，你不再持有该 Thread 的引用。另一方面，你没有办法在不同的 Activity 中对同一 Thread 进行控制。

举个例子：如果你的 Thread 需要不停地隔一段时间就要连接服务器做某种同步的话，该 Thread 需要在 Activity 没有start的时候也在运行。这个时候当你 start 一个 Activity 就没有办法在该 Activity 里面控制之前创建的 Thread。因此你便需要创建并启动一个 Service ，在 Service 里面创建、运行并控制该 Thread，这样便解决了该问题（因为任何 Activity 都可以控制同一 Service，而系统也只会创建一个对应 Service 的实例）。

因此你可以把 Service 想象成一种消息服务，而你可以在任何有 Context 的地方调用 Context.startService、Context.stopService、Context.bindService，Context.unbindService，来控制它，你也可以在 Service 里注册 BroadcastReceiver，在其他地方通过发送 broadcast 来控制它，当然这些都是 Thread 做不到的。

Service的生命周期

onCreate　　onStart　　onDestroy　　onBind

1). 被启动的服务的生命周期：如果一个Service被某个Activity 调用 Context.startService 方法启动，那么不管是否有Activity使用bindService绑定或unbindService解除绑定到该Service，该Service都在后台运行。如果一个Service被startService 方法多次启动，那么onCreate方法只会调用一次，onStart将会被调用多次（对应调用startService的次数），并且系统只会创建Service的一个实例（因此你应该知道只需要一次stopService调用）。该Service将会一直在后台运行，而不管对应程序的Activity是否在运行，直到被调用stopService，或自身的stopSelf方法。当然如果系统资源不足，android系统也可能结束服务。

2). 被绑定的服务的生命周期：如果一个Service被某个Activity 调用 Context.bindService 方法绑定启动，不管调用 bindService 调用几次，onCreate方法都只会调用一次，同时onStart方法始终不会被调用。当连接建立之后，Service将会一直运行，除非调用Context.unbindService 断开连接或者之前调用bindService 的 Context 不存在了（如Activity被finish的时候），系统将会自动停止Service，对应onDestroy将被调用。

3). 被启动又被绑定的服务的生命周期：如果一个Service又被启动又被绑定，则该Service将会一直在后台运行。并且不管如何调用，onCreate始终只会调用一次，对应startService调用多少次，Service的onStart便会调用多少次。调用unbindService将不会停止Service，而必须调用 stopService 或 Service的 stopSelf 来停止服务。

4). 当服务被停止时清除服务：当一个Service被终止（1、调用stopService；2、调用stopSelf；3、不再有绑定的连接（没有被启动））时，onDestroy方法将会被调用，在这里你应当做一些清除工作，如停止在Service中创建并运行的线程。

特别注意：

1、你应当知道在调用 bindService 绑定到Service的时候，你就应当保证在某处调用 unbindService 解除绑定（尽管 Activity 被 finish 的时候绑定会自动解除，并且Service会自动停止）；

2、你应当注意 使用 startService 启动服务之后，一定要使用 stopService停止服务，不管你是否使用bindService；

3、同时使用 startService 与 bindService 要注意到，Service 的终止，需要unbindService与stopService同时调用，才能终止 Service，不管 startService 与 bindService 的调用顺序，如果先调用 unbindService 此时服务不会自动终止，再调用 stopService 之后服务才会停止，如果先调用 stopService 此时服务也不会终止，而再调用 unbindService 或者 之前调用 bindService 的 Context 不存在了（如Activity 被 finish 的时候）之后服务才会自动停止；

4、当在旋转手机屏幕的时候，当手机屏幕在“横”“竖”变换时，此时如果你的 Activity 如果会自动旋转的话，旋转其实是 Activity 的重新创建，因此旋转之前的使用 bindService 建立的连接便会断开（Context 不存在了），对应服务的生命周期与上述相同。

5、在 sdk 2.0 及其以后的版本中，对应的 onStart 已经被否决变为了 onStartCommand，不过之前的 onStart 任然有效。这意味着，如果你开发的应用程序用的 sdk 为 2.0 及其以后的版本，那么你应当使用 onStartCommand 而不是 onStart。

下面开始上一个很简单的代码哈~里头的注释也要注意哦，有在上面没有讲到的会在注释里提到哇（尤其适用Bind方法的时候的数据传输哇）~

首先，因为要再Manifest文件里对服务进行注册，所以就先来Manifest的代码吧~

[html] view plain copy print ?

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.localservice"android:versionCode="1"
    android:versionName="1.0">
    <uses-sdkandroid:minSdkVersion="8"/>

    <applicationandroid:icon="@drawable/icon"android:label="@string/app_name">
        <activityandroid:name=".LocalServiceTestActivity"
            android:label="@string/app_name">
            <intent-filter>
                <actionandroid:name="android.intent.action.MAIN"/>
                <categoryandroid:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <serviceandroid:name=".MyService">
            <intent-filter>
                <actionandroid:name="com.test.SERVICE_TEST"/>
                <categoryandroid:name="android.intent.category.default"/>
            </intent-filter>
        </service>
    </application>
</manifest>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.test.localservice" android:versionCode="1"
      android:versionName="1.0">
      <uses-sdk android:minSdkVersion="8" />

      <application android:icon="@drawable/icon" android:label="@string/app_name">
          <activity android:name=".LocalServiceTestActivity"
                android:label="@string/app_name">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
```

```
        </activity>
        <service android:name=".MyService">
            <intent-filter>
                <action android:name="com.test.SERVICE_TEST" />
                <category android:name="android.intent.category.default" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

然后然后，是服务实现类~

[java] view plain copy print ?

```
1.
```

[java] view plain copy print ?

```
1.  package com.test.service;
2.
3.  import android.app.Service;
4.  import android.content.Intent;
5.  import android.os.Binder;
6.  import android.os.IBinder;
7.  import android.util.Log;
8.
9.  publicclass MyService extends Service {
10.
11.     publicclass LocalBinder extends Binder {
12.         String stringToSend = "I'm the test String";
13.         MyService getService() {
14.             Log.i("TAG", "getService ---> " + MyService.this);
15.             return MyService.this;
16.         }
17.     }
18.
19.     privatefinal IBinder mBinder = new LocalBinder();
20.
21.     @Override
22.     public IBinder onBind(Intent intent) {
23.         // TODO Auto-generated method stub
24.         Log.i("TAG", "onBind~~~~~~~~~~~~");
25. //      IBinder myIBinder = null;
26. //      if ( null == myIBinder )
27. //          myIBinder = new LocalBinder() ;
```

```
//      return myIBinder;
        return mBinder;     //也可以像上面几个语句那样重新new一个IBinder
        //如果这边不返回一个IBinder的接口实例，那么ServiceConnection中的onServiceConnected就不会被调用
        //那么bind所具有的传递数据的功能也就体现不出来~\(≧▽≦)/~啦啦啦（这个返回值是被作为onServiceConnected中的第二个参数的）
    }

    @Override
    publicvoid onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();

        Log.i("TAG", "onCreate~~~~~~~~~~");
    }

    @Override
    publicvoid onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
        Log.i("TAG", "onDestroy~~~~~~~~~~");
    }

    @Override
    publicvoid onStart(Intent intent, int startId) {
        // TODO Auto-generated method stub
        super.onStart(intent, startId);
        Log.i("TAG", "onStart~~~~~~");
    }

    @Override
    publicint onStartCommand(Intent intent, int flags, int startId) {
        // TODO Auto-generated method stub
        Log.i("TAG", "onStartCommand~~~~~~~~~~~~");
        returnsuper.onStartCommand(intent, flags, startId);
    }

    @Override
    publicboolean onUnbind(Intent intent) {
        // TODO Auto-generated method stub
        Log.i("TAG", "onUnbind~~~~~~~~~~~~~~~~");
        returnsuper.onUnbind(intent);
    }
}
```

package com.test.service;

import android.app.Service;
import android.content.Intent;

```
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {

    public class LocalBinder extends Binder {
        String stringToSend = "I'm the test String";
        MyService getService() {
            Log.i("TAG", "getService ---> " + MyService.this);
            return MyService.this;
        }
    }

    private final IBinder mBinder = new LocalBinder();

    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        Log.i("TAG", "onBind~~~~~~~~~~~~");
//        IBinder myIBinder = null;
//        if ( null == myIBinder )
//            myIBinder = new LocalBinder() ;
//        return myIBinder;
        return mBinder;         //也可以像上面几个语句那样重新new一个IBinder
        //如果这边不返回一个IBinder的接口实例，那么ServiceConnection中的onServiceConnected就不会被调用
        //那么bind所具有的传递数据的功能也就体现不出来~\(≧▽≦)/~啦啦啦（这个返回值是被作为onServiceConnected中的第二个参数的）
    }

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();

        Log.i("TAG", "onCreate~~~~~~~~~~");
    }

    @Override
    public void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
        Log.i("TAG", "onDestroy~~~~~~~~~~~");
    }

    @Override
    public void onStart(Intent intent, int startId) {
        // TODO Auto-generated method stub
        super.onStart(intent, startId);
        Log.i("TAG", "onStart~~~~~~");
```

28. C# Serializable对象序列化的作用(8731)
29. Java 使用executeUpdate向数据库中创建表格(8608)
30. C++ 使用API写Windows程序(8398)
31. WPF 获得DataTemplate中的控件(7857)
32. C++ AFX_MANAGE_STATE(AfxGetStaticMo的作用(7732)
33. css <meta name="viewport" content="xx">(7452)
34. java实现文件单词频率统计(7216)
35. 创建Android的Hello World应用程序(7077)
36. WPF Button 透明效果(7024)
37. Android 隐藏Fragment(6959)
38. word 2007中在页眉中插入或这删除下划线(6708)
39. WPF 控件使用之ComboBox(6393)
40. WPF 附加属性(6165)

评论排行榜

1. WPF 一个MVVM的简单例子(12)
2. C# Dictionary用法总结(8)
3. Android中Service的使用详解和注意点（LocalService）(8)
4. html5 图片旋转(6)
5. C#操作xml SelectNodes,SelectSingleNode总是返回NULL 与 xPath 介绍(5)

推荐排行榜

1. C# Dictionary用法总结(15)
2. Android中Service的使用详解和注意点（LocalService）(5)
3. wpf 多线程(5)
4. WPF 附加属性(4)
5. WPF ListBox(4)
6. Metro Style App开发快速入门 之文件访问操作示例(4)

```
        }

        @Override
        public int onStartCommand(Intent intent, int flags, int startId) {
            // TODO Auto-generated method stub
            Log.i("TAG", "onStartCommand~~~~~~~~~~~~");
            return super.onStartCommand(intent, flags, startId);
        }

        @Override
        public boolean onUnbind(Intent intent) {
            // TODO Auto-generated method stub
            Log.i("TAG", "onUnbind~~~~~~~~~~~~~~~~");
            return super.onUnbind(intent);
        }
    }
```

再来，就是我们的Activity的测试类啦~

[java] view plain copy print ?

```
<pre class="java" name="code">package com.test.service;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass ServiceTestActivity extends Activity {
    private Button startButton, bindButton;
    private Button stopButton, unbindButton;
    private ServiceConnection sc;
    private MediaPlayer mediaPlayer = null;
    private MyService myService;// 类似于MediaPlayer mPlayer = new
                                // MediaPlayer();只不过这边的服务是自定义的，不是系统提供好了的

    /** Called when the activity is first created. */
```

```
25.     @Override
26.     publicvoid onCreate(Bundle savedInstanceState) {
27.         super.onCreate(savedInstanceState);
28.         setContentView(R.layout.main);
29.
30.         startButton = (Button) findViewById(R.id.startbutton_id);
31.         stopButton = (Button) findViewById(R.id.stopbutton_id);
32.         bindButton = (Button) findViewById(R.id.bindbutton_id);
33.         unbindButton = (Button) findViewById(R.id.unbindbutton_id);
34.
35.         sc = new ServiceConnection() {
36.             /*
37.              * 只有在MyService中的onBind方法中返回一个IBinder实例才会在Bind的时候
38.              * 调用onServiceConnection回调方法
39.              * 第二个参数service就是MyService中onBind方法return的那个IBinder实例，可以利用这个来传递数据
40.              */
41.             @Override
42.             publicvoid onServiceConnected(ComponentName name, IBinder service) {
43.                 // TODO Auto-generated method stub
44.                 myService = ((MyService.LocalBinder) service).getService();
45.                 String recStr = ((MyService.LocalBinder) service).stringToSend;
46.                 //利用IBinder对象传递过来的字符串数据（其他数据也可以啦，哪怕是一个对象也OK~~）
47.                 Log.i("TAG","The String is : " + recStr);
48.                 Log.i("TAG", "onServiceConnected : myService ---> " + myService);
49.             }
50.
51.             @Override
52.             publicvoid onServiceDisconnected(ComponentName name) {
53.                 /* SDK上是这么说的：
54.                  * This is called when the connection with the service has been unexpectedly disconnected
55.                  * that is, its process crashed. Because it is running in our same process, we should never see this happen.
56.                  * 所以说，只有在service因异常而断开连接的时候，这个方法才会用到*/
57.                 // TODO Auto-generated method stub
58.                 sc = null;
59.                 Log.i("TAG", "onServiceDisconnected : ServiceConnection --->"
60.                         + sc);
61.             }
62.
63.         };
64.         startButton.setOnClickListener(new OnClickListener() {
65.
66.             @Override
67.             publicvoid onClick(View v) {
68.                 // TODO Auto-generated method stub
69.                 Intent intent = new Intent(ServiceTestActivity.this,
70.                         MyService.class);
```

```
                startService(intent);
                Log.i("TAG", "Start button clicked");
            }
        });

        stopButton.setOnClickListener(new OnClickListener() {

            @Override
            publicvoid onClick(View v) {
                // TODO Auto-generated method stub

                /*
                 * Intent intent = new
                 * Intent(LocalServiceTestActivity.this,MyService.class);
                 * stopService(intent); 这种方法也是可以的哈~
                 */

                Intent intent = new Intent();
                intent.setAction("com.test.SERVICE_TEST");
                stopService(intent);
                Log.i("TAG", "Stop Button clicked");
            }
        });

        bindButton.setOnClickListener(new OnClickListener() {

            @Override
            publicvoid onClick(View v) {
                // TODO Auto-generated method stub
//              Intent intent = new Intent(LocalServiceTestActivity.this,
//                      MyService.class);//这样也可以的
                Intent intent = new Intent();
                intent.setAction("com.test.SERVICE_TEST");
                bindService(intent, sc, Context.BIND_AUTO_CREATE);//bind多次也只会调用一次onBind方法
                Log.i("TAG", "Bind button clicked");
            }
        });

        unbindButton.setOnClickListener(new OnClickListener() {

            @Override
            publicvoid onClick(View v) {
                // TODO Auto-generated method stub
                unbindService(sc);
                // 这边如果重复unBind会报错，提示该服务没有注册的错误——IllegalArgumentException:
                // Service not registered: null
```

```
                // 所以一般会设置一个flag去看这个service
                // bind后有没有被unBind过，没有unBind过才能调用unBind方法(这边我就不设置了哈~\(≧▽≦)/~啦啦啦)
                Log.i("TAG", "Unbind Button clicked");
            }
        });
    }
}</pre><br>
<br>
<pre></pre>
<br>
相信开头的介绍和代码里的注释应该对大家理解和使用Service有所帮助哈~不过这边就先只讲LocalService吧，剩下的RemoteService就等下次在说喽~
<p></p>
<p><span style="white-space: pre;"><span style="line-height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span
style="line-height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span style="line-height: 24px; font-family:
verdana,宋体,Arial; font-size: 13px;"><span style="white-space: pre;"><span style="line-height: 24px; font-family: verdana,宋
体,Arial; font-size: 13px;"><span style="line-height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span style="line-
height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span style="white-space: pre;"><span style="line-height: 24px;
font-family: verdana,宋体,Arial; font-size: 13px;"><span style="line-height: 24px; font-family: verdana,宋体,Arial; font-size:
13px;"><span style="white-space: pre;"><span style="line-height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span
style="color: rgb(51, 51, 51); line-height: 24px; font-family: verdana,宋体,Arial; font-size: 13px;"><span style="white-space:
pre;"></span></span></span></span></span></span></span></span></span></span></span></span></span></p>
```

Android服务总结

Android服务使用

Android中Service的使用详解和注意点（LocalService）

Android 使用AIDL调用外部服务 （RemoteService）

Android 中的 Service 全面总结

分类: S.Android

« 上一篇：ContentProvider 使用示例（转载）
» 下一篇：linux ubuntu系统下，adb不是内部命令 （如何才能让adb命令可以使用）

Feedback

#1楼
2014-11-01 18:21 by climberDing

好文章啊

支持(0) 反对(0)

#2楼
2014-12-30 14:52 by sangxb

好文要顶！

支持(0) 反对(0)

#3楼
2015-01-26 16:24 by 黑夜孤灯

下面这个描述不正确吧：

当在旋转手机屏幕的时候，当手机屏幕在“横”“竖”变换时，此时如果你的 Activity 如果会自动旋转的话，旋转其实是 Activity 的重新创建，因此旋转之前的使用 bindService 建立的连接便会断开（Context 不存在了），对应服务的生命周期与上述相同。

自动旋转和activity重新创建能关联上这也太不靠谱

支持(0) 反对(2)

#4楼
2015-03-21 15:05 by 二向箔

@ 黑夜孤灯
在大部分情况下，自动旋转的确是会重新创建activity的。

支持(0) 反对(0)

#5楼
2015-04-12 12:54 by 明天+=灿烂

好文要顶

支持(0) 反对(0)

#6楼
2015-09-18 17:00 by mmbbjjcc

内容很详细，学习了，谢谢

支持(0) 反对(0)

#7楼
2015-09-22 14:22 by drummor

当被启动又被绑定的服务的生命周期，如果是直接解除绑定的话Service就直接销毁了，并不需要stop啊

支持(0) 反对(1)

#8楼
2016-06-16 20:33 by 放纵的卡尔

@ drummor
没启动服务的话是直接销毁！但是启动了服务的话，解除绑定只是unbind，并没有执行ondestroy。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 人类太多余？且慢，先听AI科学家详解AlphaGo Zero的伟大与局限
· 比特币价格首破6000美元 其创造者持币价值59亿美元
· 亚马逊要建第二总部，引发了100+选手参战的城市战争
· 传永安行将收购小蓝单车，千万“救助款”已到账
· Intel发布开源增强学习框架Coach
» 更多新闻...

最新知识库文章:
· 实用VPC虚拟私有云设计原则
· 如何阅读计算机科学类的书
· Google 及其云智慧
· 做到这一点，你也可以成为优秀的程序员
· 写给立志做码农的大学生
» 更多知识库文章...