# Service定义

Service（服务）是一个可以在后台执行长时间操作而不提供用户界面的应用组件。服务可由其他应用组件启动，而且即使用户切换到其他应用，服务仍会在后台继续运行。此外，应用可以绑定到服务，以与之进行交互，甚至是执行进程间通信（IPC）。

服务基本上分为两种形式：

（1）启动。当应用组件通过调用startService()启动服务时，服务即处于启动状态。一旦启动，服务就可在后台无限期运行，即使调用它的组件已被销毁也不受影响。已启动的服务一般是执行单一操作，而且不会将结果返回给调用方。例：下载或上传文件。

（2）绑定。当应用组件通过调用bindService()绑定到服务时，服务即处于绑定状态。绑定服务允许组件与服务进行交互、发送请求、获取结果，甚至是利用进程间通信跨进程执行这些操作。仅当与另一个应用组件绑定时，绑定服务才会运行。多个组件可以同时绑定到一个服务，但全部取消绑定后，该服务即会被销毁。

一个服务可以同时以这两种方式运行，既可以是启动服务（无限期运行），也允许绑定。

注意：服务在其托管进程的主线程内运行，它既不创建自己的线程，也不在单独的进程中运行，这意味着如果想要让服务执行某些耗时长的操作，则应通过在服务中创建新线程来实现。这样做既能降低ANR（应用未响应）的风险，也能保证Activity与用户的交互。

# Service主要回调方法

要创建服务，必须创建Service的子类（或是使用它的一个现有子类），并重写它的一部分回调方法。以下为所有Service通用的，必须重写的最重要的回调方法：

onStartCommand()：当另一个应用组件（通常是Activity）通过调用Context.startService()请求启动服务时，系统将调用该方法。该方法一旦被调用，服务即可在后台无限期运行。因此，在服务工作完成后，需要显式调用stopSelf()或Context.stopService()来停止服务。

onStartCommand()方法必须返回一个特定常量（IntentService的默认实现已经处理了这个问题，不过也可以修改），如下：

START\_NOT\_STICKY：如果系统在onStartCommand()返回后终止服务，则除非有挂起Intent要传递，否则系统不会重建服务。

START\_STICKY：如果系统在onStartCommand()返回后终止服务，则会重建服务并调用onStartCommand()，但不会重新传递最后一个Intent。除非有挂起Intent要启动服务（此时会传递这些Intent），否则系统会通过空Intent调用onStartCommand()。这适用于无限期运行等待命令的服务。

START\_REDELIVER\_INTENT：如果系统在onStartCommand()返回后终止服务，则会重建服务并调用onStartCommand()，并重新传递最后一个Intent。这适用于主动执行应该立即恢复的作业（如下载）的服务。

其他常量可阅读<https://developer.android.google.cn/reference/android/app/Service.html#stopSelfResult(int)>

onBind()：当另一个组件想通过调用bindService与服务绑定时，系统将调用该方法。在此方法的实现中，您必须通过返回IBinder提供一个接口，供客户端与服务进行通信。如果不希望服务允许绑定，则应返回null。

onCreate()：首次创建服务时，系统将调用此方法来执行一次性设置程序（在onStartCommand()与onBind()前）。如果服务已在运行，则不会调用该方法。

onDestroy()：当服务不再被使用且将被销毁时，系统将调用此方法。服务应该实现该方法来清理所有使用过的资源，如线程、侦听器等。这是服务最后一个调用。

如果选择扩展IntentService，那么除onBind()与onHandleIntent()外，实现其他方法时必须调用超类实现。

# 声明服务

和Activity一样，在使用Service之前，必须在应用的manifest文件中声明，通过添加<service>元素作为<application>元素的子元素，并设置android:name属性（必需）。此外，还可以通过添加android:exported并设置为”false”来确保服务仅运用与本应用，阻止其他应用启动该服务。

# 向用户发送通知

一旦服务运行起来，服务即可通过Toast通知或是状态栏通知来通知用户所发生的事件。通常，当某些后台工作已经完成（例如文件下载完成）且需要用户对其进行操作时，状态栏通知是最佳方法（例如查看已下载的文件）。

# 前台服务

前台服务被认为是用户主动意识到的一种服务，因而即便内存不足，系统也不会考虑将其终止。前台服务必须为状态栏提供通知，放在“正在进行”下方，这意味着除非服务停止或从前台移除，否则不能清除通知。

# Service生命周期



无论服务是通过 startService() 还是 bindService() 创建，都会为所有服务调用 onCreate() 和 onDestroy() 方法。

服务的整个生命周期从调用 onCreate() 开始起，到 onDestroy() 返回时结束。与 Activity 类似，服务也在 onCreate() 中完成初始设置，并在 onDestroy() 中释放所有剩余资源。例如，音乐播放服务可以在 onCreate() 中创建用于播放音乐的线程，然后在 onDestroy() 中停止该线程。

服务的有效生命周期从调用 onStartCommand() 或 onBind() 方法开始。每种方法均有 {Intent 对象，该对象分别传递到 startService() 或 bindService()。服务生命周期（从创建到销毁）可以遵循两条不同的路径：

启动服务：该服务在其他组件调用 startService() 时创建，然后无限期运行，且必须通过调用 stopSelf() 来自行停止运行。此外，其他组件也可以通过调用 stopService() 来停止服务。服务停止后，系统会将其销毁。

绑定服务：该服务在另一个组件（客户端）调用 bindService() 时创建。然后，客户端通过 IBinder 接口与服务进行通信。客户端可以通过调用 unbindService() 关闭连接。多个客户端可以绑定到相同服务，而且当所有绑定全部取消后，系统即会销毁该服务。 （服务不必自行停止运行。）

这两条路径并非完全独立。也就是说，您可以绑定到已经使用 startService() 启动的服务。例如，可以通过使用 Intent（标识要播放的音乐）调用 startService() 来启动后台音乐服务。随后，可能在用户需要稍加控制播放器或获取有关当前播放歌曲的信息时，Activity 可以通过调用 bindService() 绑定到服务。在这种情况下，除非所有客户端均取消绑定，否则 stopService() 或 stopSelf() 不会实际停止服务。

尽管启动服务是通过调用 stopSelf() 或 stopService() 来停止，但是该服务并无相应的回调（没有 onStop() 回调）。因此，除非服务绑定到客户端，否则在服务停止时，系统会将其销毁 — onDestroy() 是接收到的唯一回调。