# 基础知识

绑定服务是 Service 类的实现，可让其他应用与其绑定和交互。要提供服务绑定，您必须实现 onBind() 回调方法。该方法返回的 IBinder 对象定义了客户端用来与服务进行交互的编程接口。可以创建同时具有已启动和绑定两种状态的服务。也就是说，可通过调用 startService() 启动该服务，让服务无限期运行；此外，还可通过调用 bindService() 使客户端绑定到服务。

当最后一个客户端取消与服务的绑定时，系统会将服务销毁（除非 startService() 也启动了该服务）。如果服务同时具有已启动和绑定状态，则服务启动后，系统“不会”在所有客户端都取消绑定时销毁服务。 为此，必须通过调用 stopSelf() 或 stopService() 显式停止服务。

多个客户端可同时连接到一个服务。不过，只有在第一个客户端绑定时，系统才会调用服务的 onBind() 方法来检索 IBinder。系统随后无需再次调用 onBind()，便可将同一 IBinder 传递至任何其他绑定的客户端。

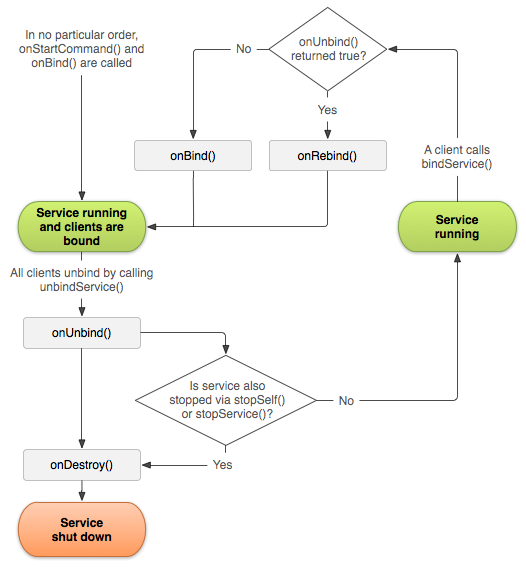
客户端可通过调用 bindService() 绑定到服务。调用时，它必须提供 ServiceConnection 的实现，后者会监控与服务的连接。bindService() 方法会立即无值返回，但当 Android 系统创建客户端与服务之间的连接时，会对 ServiceConnection 调用 onServiceConnected()，向客户端传递用来与服务通信的 IBinder。

# 绑定服务的生命周期

当服务与所有客户端之间的绑定全部取消时，Android 系统便会销毁服务（除非还使用 onStartCommand() 启动了该服务）。因此，服务是纯粹的绑定服务，则无需对其生命周期进行管理。

不过，如果您选择实现 onStartCommand() 回调方法，则您必须显式停止服务，因为系统现在已将服务视为已启动。在此情况下，服务将一直运行到其通过 stopSelf() 自行停止，或其他组件调用 stopService() 为止，无论其是否绑定到任何客户端。

此外，如果您的服务已启动并接受绑定，则当系统调用您的 onUnbind() 方法时，如果您想在客户端下一次绑定到服务时接收 onRebind() 调用，则可选择返回 true。onRebind() 返回空值，但客户端仍在其 onServiceConnected() 回调中接收 IBinder。



# 创建绑定服务

创建提供绑定的服务时，必须提供IBinder，用以提供客户端用来与服务进行交互的编程接口。有以下三种方式：

（1）扩展Binder类

如果服务仅供本地应用使用，不需要跨进程工作，则可以实现Binder类，让客户端通过该类直接访问服务中的公共方法。这种方法要求客户端与服务位于同一应用和进程内。这也是最常见的一种情况。具体方法如下：

1、在服务中创建一个满足以下要求之一的Binder实例：

* 包含客户端可调用的公共方法
* 返回当前Service实例，其中包含客户端可调用的公共方法
* 返回由服务承载的其他类的实例，其中包含客户端可使用的公共方法

2、在onBind()回调方法中返回此Binder实例。

3、客户端中从onServiceConnected()中接收Binder，并使用提供的方法调用绑定服务。

（2）使用Messenger

如果服务需要与远程进程通信，且不需要执行多线程操作，则可使用Messenger为服务提供接口。具体方法如下：

1、服务实现一个Handler，由其接收来自客户端的每个调用的回调。

2、用该Handler的一个实例创建Messenger对象。

3、Messenger创建一个IBinder（一般通过Messenger的getBinder()方法），服务通过onBind()使其返回客户端。

4、客户端在onServiceConnected()使用参数IBinder将Messenger（引用服务的Handler）实例化，然后使用后者发送消息给服务。

5、服务在其Handler中处理消息。

如果想让服务作出响应，则还需要在客户端创建Messenger对象，向服务发送Message时将其replyTo域设置为客户端的Messenger对象。

（3）AIDL

仅在跨进程通信且需要执行多线程操作时使用。

# 绑定到服务

应用组件通过boolean bindService (Intent service, ServiceConnection conn, int flags)绑定到服务。

Intent service：命名绑定服务的Intent对象。

ServiceConnection conn：实现了ServiceConnection的对象。必须重写两个回调方法：onServiceConnected()（连接成功时调用）与onServiceDisconnected()（与服务的连接意外中断时，注意客户端取消绑定时不会调用）。

int flags：一个指示绑定选项的标志。它通常应该是BIND\_AUTO\_CREATE，以便创建尚未激活的服务。以下为其他参数说明：

* BIND\_ABOVE\_CLIENT：说明所要启动的服务比应用本身更加重要，暗示系统的内存回收机制应当尽量在回收应用之后才回收该服务（不保证）。
* BIND\_ADJUST\_WITH\_ACTIVITY：服务重要程度随Activity的可见程度变化。
* BIND\_ALLOW\_OOM\_MANAGEMENT（不确定）：服务允许在系统OOM时被杀死。
* BIND\_AUTO\_CREATE（最常用）：在服务被绑定时自动创建。
* BIND\_DEBUG\_UNBIND：仅供测试用。当发生不正确的绑定时打印解绑命令集。
* BIND\_EXTERNAL\_SERVICE：服务来自外部。使用该标志后服务的代码将在呼叫服务的应用的包内。
* BIND\_IMPORTANT：服务对客户端极其重要。
* BIND\_NOT\_FOREGROUND：禁止将服务所属的进程置于前台。
* BIND\_WAIVE\_PRIORITY：将服务所属进程视作后台任务。

# 附加说明

* 与服务通信的方法中应该始终捕获 DeadObjectException 异常，它们是在连接中断时引发的。这是远程方法引发的唯一异常。
* 如果只需要在 Activity 可见时与服务交互，则应在 onStart() 期间绑定，在 onStop() 期间取消绑定。
* 如果希望 Activity 在后台停止运行状态下仍可接收响应，则可在 onCreate() 期间绑定，在 onDestroy() 期间取消绑定。请注意，这意味着Activity 在其整个运行过程中（甚至包括后台运行期间）都需要使用服务，因此如果服务位于其他进程内，那么当您提高该进程的权重时，系统终止该进程的可能性会增加。
* 通常情况下，切勿在 Activity 的 onResume() 和 onPause() 期间绑定和取消绑定，因为每一次生命周期转换都会发生这些回调，您应该使发生在这些转换期间的处理保持在最低水平。此外，如果您的应用内的多个 Activity 绑定到同一服务，并且其中两个 Activity 之间发生了转换，则如果当前 Activity 在下一个 Activity 绑定（恢复期间）之前取消绑定（暂停期间），系统可能会销毁服务并重建服务。 （Activity文档中介绍了这种有关 Activity 如何协调其生命周期的 Activity 转换。）