# 创建启动服务

应用组件通过startService()并传递Intent对象（指定服务并包含待使用服务的所有数据）来启动服务，服务通过onStartCommand()接收Intent。

可以通过扩展两个类来创建自定义服务：

（1）IntentService：Service的子类，使用工作线程逐一处理所有启动请求。如果不要求服务同时处理多个请求，这是最好的选择，只需实现onHandIntent()方法即可。

IntentService会执行以下操作：

* + 在主线程外创建默认的工作线程，用于传递给onStartCommand()的所有Intent。
  + 创建工作队列，用于将Intent逐一传递给onHandleIntent()实现，因此永远不用考虑多线程问题。
  + **在处理完所有启动请求后自动停止服务，因此永远不用调用stopSelf()。**
  + 提供onBind()的默认实现（返回null）。
  + 提供onStartCommand()的默认实现（将Intent依次发送到工作队列）。

综上所述，在通过扩展IntentService来创建服务的过程中，必需的步骤只有提供默认构造器以及实现onHandleIntent()方法。如果还需要重写其他回调方法（如 onCreate()、onStartCommand() 或 onDestroy()），要确保调用超类实现，以便 IntentService 能够妥善处理工作线程的生命周期。

（2）Service：若要求服务执行多线程操作，则可通过扩展Service类来处理每个Intent。需要手动实现onStartCommand()方法处理任务，并在任务完成时调用stopSelf()结束服务。通常做法是在onStartCommand()中为每个任务创建单独的线程并运行。

# 启动服务

通过将Intent传递给Content.startService()启动服务，由Android系统调用服务的onStartCommand()方法。如果服务尚未运行，则会先执行onCreate()方法。

如果服务亦未提供绑定，则使用 startService() 传递的 Intent 是应用组件与服务之间唯一的通信模式。但是，如果您希望服务返回结果，则启动服务的客户端可以为广播创建一个 PendingIntent （使用 getBroadcast()），并通过启动服务的 Intent 传递给服务。然后，服务就可以使用广播传递结果。

# 停止服务

多个服务启动请求会导致对服务的onStartCommand()方法的多次调用。但是，要停止服务，只需一个服务停止请求（stopSelf()或Context.stopService()）即可。

需要注意的是，如果服务同时处理多个onStartCommand()请求，则不应当在处理完一个请求后销毁服务，因为这时服务可能已经收到了其他请求。为此，可以使用stopSelf(int)确保服务停止请求基于最近的启动请求。具体方法是调用stopSelf()时传递启动请求的ID。这样做的话，如果在调用stopSelf()之前收到了新请求，ID就不匹配，服务也不会终止。