Service

1,Service 是一个可以在后台执行长时间运行操作而不使用用户界面的应用组件。
2,组件可以绑定到服务，以与之进行交互，甚至是执行进程间通信 (IPC)。
3,服务可以处理网络事务、播放音乐，执行文件 I/O 或与内容提供程序交互，而所有这一切均可在后台进行。

服务两种形式：

启动和绑定

startService()； bindService()

创建私有服务

注意：

服务在其托管进程的主线程中运行，它既不创建自己的线程，也不在单独的进程中运行（除非另行指定）。这意味着，如果服务将执行任何 CPU 密集型工作或阻止性操作（例如 MP3 播放或联网），则应在服务内创建新线程来完成这项工作。通过使用单独的线程，可以降低发生“应用无响应”(ANR) 错误的风险，而应用的主线程仍可继续专注于运行用户与 Activity 之间的交互。

服务与线程：

简单地说，服务是一种即使用户未与应用交互，但它仍可以在后台运行的组件。

如需在主线程外部执行工作，不过只是在用户正在与应用交互时才有此需要，则应创建新线程而非服务

例如，如果您只是想在 Activity 运行的同时播放一些音乐，则可在 onCreate() 中创建线程，在 onStart() 中启动线程，然后在 onStop() 中停止线程。您还可以考虑使用 AsyncTask 或HandlerThread，而非传统的 Thread类。

如果您确实要使用服务，则默认情况下，它仍会在应用的主线程中运行，因此，如果服务执行的是密集型或阻止性操作，则您仍应在服务内创建新线程。

规定服务启动权限

android:name 属性是唯一必需的属性，用于指定服务的类名。应用一旦发布，即不应更改此类名，如若不然，可能会存在因依赖显式 Intent 启动或绑定服务而破坏代码的风险

为了确保应用的安全性，**请始终使用显式 Intent 启动或绑定 Service**，且不要为服务声明 Intent 过滤器。启动哪个服务存在一定的不确定性，而如果对这种不确定性的考量非常有必要，则可为服务提供 Intent 过滤器并从 Intent 中排除相应的组件名称，但随后必须使用 setPackage() 方法设置 Intent 的软件包，这样可以充分消除目标服务的不确定性。

此外，还可以通过添加 android:exported 属性并将其设置为 "false"，确保服务仅适用于您的应用。这可以有效阻止其他应用启动您的服务，即便在使用显式 Intent 时也如此。

前台运行服务

服务的**整个生命周期**从调用 onCreate() 开始起，到 onDestroy() 返回时结束。
服务的**有效生命周期**从调用 onStartCommand() 或 onBind() 方法开始。

类别	区别	优点	缺点	应用
本地服务（local service）	该服务依附在主进程上	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。	非常常见的应用如：HTC的音乐播放服务，天天动听音乐播放服务。

远程服务（Remote）	该服务是独立的进程	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。	一些提供系统服务的Service，这种Service是常驻的。

remote服务还是很少见的，并且一般都是系统服务。

前台服务	会在通知一栏显示 ONGOING 的 Notification，	当服务被终止的时候，通知一栏的 Notification 也会消失，这样对于用户有一定的通知作用。常见的如音乐播放服务。
后台服务	默认的服务即为后台服务，即不会在通知一栏显示 ONGOING 的 Notification。	当服务被终止的时候，用户是看不到效果的。某些不需要运行或终止提示的服务，如天气更新，日期同步，邮件同步等。

备注:
　有同学可能会问，后台服务我们可以自己创建 ONGOING 的 Notification 这样就成为前台服务吗？答案是否定的，前台服务是在做了上述工作之后需要调用 startForeground （android 2.0 及其以后版本）或 setForeground （android 2.0 以前的版本）使服务成为 前台服务。这样做的好处在于，当服务被外部强制终止掉的时候，ONGOING 的 Notification 任然会移除掉。

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
bindService 启动的服务	该方法启动的服务要进行通信。停止服务使用unbindService
startService 同时也 bindService 启动的服务	停止服务应同时使用stepService与unbindService

注意:
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。

service与Activity相连接:
service里创建一个公开类：Binder extends android.os.Binder
IBinder new Bind里面返回new Binder
在Binder里面书写方法