Service 基础知识

一、什么是 Android 中的 service？

Service 是 Android 中一个类，它是 Android 四大组件之一，使用 Service 可以在后台执行耗时的操作(注意需另启子线程)，其中 Service 并不与用户产生 UI 交互。其他的应用组件可以启动 Service，即便用户切换了其他应用，启动的 Service 仍可在后台运行。一个组件可以与 Service 绑定并与之交互，甚至是跨进程通信。通常情况下 Service 可以在后台执行网络请求、播放音乐、执行文件读写操作或者与 content provider 交互等。

二、Service 如何创建和注册？

只需要定义一个类，继承 Service，或者在 Android Studio 中直接使用直接选择 Service 即可，同时注意在代码中或者 Android Manifest.xml 中注册服务。

1.创建 service 对象

```
MyService  receiver = new  MyService();
```

2.在配置文件中注册：

```
<service android:name="包名.类名" ></service>
```

3．启动服务：

```
Intent intent=new Intent(MainActivity.this,MyService.class);
startService(intent);
```

三、Service 的分类：

Android 中按照启动方式将服务分为启动服务和绑定服务两类，其中启动服务通常是 StartService()，绑定服务是 bindService()。

1. 启动服务：其他组件调用 startService()方法启动一个 Service。一旦启动，

Service 将一直运行在后台，即便启动 Service 的组件已被 destroy。但是 Service 会在后台执行单独的操作，也并不给启动它的组件返回结果。比如说，一个 start 的 Service 执行在后台下载或上传一个文件的操作，完成之后，Service 自行停止。

2. 绑定服务：其他组件调用 bindService()方法绑定一个 Service。通过绑定方式启动的 Service 是一个 client-server 结构，该 Service 可以与绑定它的组件进行交互。一个 bound service 仅在有组件与其绑定时才会运行，多个组件可与一个 service 绑定，service 不再与任何组件绑定时，该 service 会被 destroy。

四、StartService 生命周期及进程相关：

1. 客户端首次请求服务端时的生命周期方法有：onCreate() >> onStartCommand >> onDestroy()；

注 :onStartCommand(..)可以多次被调用 ,onDestroy()与 onCreate()相互匹配，当用户强制 kill 掉进程时，onDestroy()是不会执行的。

2.对于同一类型的 Service，Service 实例一次永远只存在一个，而不管 Client 是否是相同的组件，也不管 Client 是否处于相同的进程中。

3.Service 通过 startService(..)启动 Service 后 ,此时 Service 的生命周期与 Client 本身的什么周期是没有任何关系的 ,只有 Client 调用 stopService(..)或 Service 本身调用 stopSelf(..)才能停止此 Service。当然，当用户强制 kill 掉 Service 进程或系统因内存不足也可能 kill 掉此 Service。

4.Client A 通过 startService(..)启动 Service 后,可以在其他 Client(如 Client B、Client C) 通过调用 stopService(..)结束此 Service。

5.startService(Intent serviceIntent)，其中的 intent 既可以是显式 Intent，也可以是隐式 Intent，当 Client 与 Service 同处于一个 App 时，一般推荐使用显示 Intent。当处于不同 App 时，只能使用隐式 Intent。

当 Service 需要运行在单独的进程中，AndroidManifest.xml 声明时需要通过 android:process 指明此进程名称，当此 Service 需要对其他 App 开放时，android:exported 属性值需要设置为 true(当然，在有 intent-filter 时默认值就是 true)。

五、Bound Service 生命周期及进程相关：

Bound Service 的主要特性在于 Service 的生命周期是依附于 Client 的生命周期的，当 Client 不存在时，Bound Service 将执行 onDestroy，同时通过 Service 中的 Binder 对象可以较为方便进行 Client-Service 通信。Bound Service 一般使用过程如下：

a) 自定义 Service 继承基类 Service，并重写 onBind(Intent intent)方法，此方法中需要返回具体的 Binder 对象；
b) Client 通过实现 ServiceConnection 接口来自定义 ServiceConnection，并通过 bindService (Intent service, ServiceConnection sc, int flags) 方法将 Service 绑定到此 Client 上；
c) 自定义的 ServiceConnection 中实现 onServiceConnected(ComponentName name, IBinder binder)方法，获取 Service 端 Binder 实例；
d) 通过获取的 Binder 实例进行 Service 端其他公共方法的调用，以完成 Client-Service 通信；

e) 当 Client 在恰当的生命周期（如 onDestroy 等）时，此时需要解绑之前已经绑定的 Service ,通过调用函数 unbindService(ServiceConnection sc)。