Service组件启动过程

Service组件在新进程中的启动过程：

Client组件启动Server组件的过程如下：

1. Client组件向ActivityManagerService发送一个启动Server组件的进程间通信请求。
2. ActivityManagerService发现用来运行Server组件的应用程序进程不存在，因此，它就会首先将Server组件的信息保存下来，接着创建一个新的应用程序进程。
3. 新的应用程序进程启动完成之后，就会向ActivityManagerService发送一个启动完成的进程间通信请求，以便ActivityManagerService可以继续执行启动Server组件的操作。
4. ActivityManagerService将第二步保存下来的Server组件信息发送给第2步创建的应用程序进程，以便它可以将Server组件启动起来。

这一过程一共分为19个步骤：

1. onCreate函数(在onCreate函数中调用startService函数来启动ContextWrapper)
2. ContextWrapper.startService(当Activity组件启动时,ActivityThread类都会为它创建ContextImpl对象，实际上调用了ContextImpl类的成员函数startService来启动Server组件的操作)
3. ContextImpl.startService(在此函数中通过调用ActivityManagerNative类的静态成员函数getDefault来获得ActivityManagerService的一个代理对象，接着调用这个代理对象的成员函数startService来请求ActivityManagerService启动Server组件)
4. ActivityManagerProxy.startService(在startService函数中将传进来的参数写入到Parcel对象中，再通过ActivityManagerProxy类内部的一个Binder代理对象向ActivityManagerService发送一个类型为START\_SERVICE\_TRANSACTION的进程间通信请求)
5. ActivityManagerService .startService(ActivityManagerService类的成员函数startService用来处理类型为START\_SERVICE\_TRANSACTION的进程间通信请求。进一步调用另一个成员函数startServiceLocked来进一步处理刚才的进程间通信请求。)
6. ActivityManagerService.startServiceLocked(在startServiceLocked函数中首先会调用retrieveServiceLocked来查找是否存在与参数service对应的ServiceRecord组件的对象。如果不存在，回到PMS中去获取与参数service对应的一个service组件的信息，将它封装成一个ServiceRecord对象，再封装成一个ServiceLookupResult对象返回给调用者。在获得与参数service对应的ServiceRecord对象之后，调用AMS的bringUpServiceLocked来启动Service组件)
7. ActivityManagerService. bringUpServiceLocked(会对app中service的配置进行一系列的检查，如果发现在AMS中已经存在一个对应的进程，则会直接在此应用程序中启动这个Service组件。如果不存在，则会将ServiceRecord对象保存在AMS的成员变量mPendingService。)
8. ActivityManagerService.startProcessLocked(主要是调用Process类的静态成员函数start来创建一个新的应用程序进程。)
9. ActivityThread.main(为创建应用程序进程的入口函数，会创建一个ActivityThread对象和一个ApplicationThread对象)
10. ActivityManagerProxy.attachApplication(会向AMS发送一个类型为ATTACH\_APPLICATION\_TRANSACTION的进程间通信请求，并将之前创建的ApplicationThread对象传递给AMS)
11. ActivityManagerService.attachApplication(调用AMS的成员函数attachApplicationLocked来处理类型为ATTACH\_APPLICATION\_TRANSACTION的进程间通信请求)
12. ActivityManagerService. attachApplicationLocked(在此函数中会进行一系列检查，在检查完毕之后会调用AMS的成员函数realStartServiceLocked来启动保存在成员变量mPendingService的ServiceRecord对象)
13. ActivityManagerService. realStartServiceLocked(在此函数中调用ActivityThread对象的成员函数scheduleCreateService来请求在新创建的应用程序进程中将service组件启动起来)。
14. ApplicationThreadProxy. scheduleCreateService(调用此函数向应用程序进程发送一个类型为SCHEDULE\_CREATE\_SERVICE\_TRANSACTION的进程间通信请求)
15. ApplicationThread. scheduleCreateService(调用此函数来处理类型为SCHEDULE\_CREATE\_SERVICE\_TRANSACTION的进程间通信请求，并将要启动的service组件的信息封装成一个CreateServiceData对象，再调用此类的成员函数将其封装成一个CREATE\_SERVICE的消息，以便能够发送到应用进程的消息队列中)
16. ApplicationThread.queueOrSendMessage(将上一步封装的消息在此函数中调用ActivityThread的成员函数将其发送到应用进程的消息队列中)
17. H.handleMessage(在此函数中将service组件的信息从封装中取出来，调用ActivityThread的成员函数handleCreateService将service启动起来)
18. ActivityThread. handleCreateService(在此函数中完成service对象的创建，并在最后调用oncreate函数)
19. 启动完成后将其注册到ServiceManager中