**Android service机制**

参考：<https://blog.csdn.net/Luoshengyang/article/details/6677029>

<https://blog.csdn.net/Luoshengyang/article/details/6745181>

1. startService
2. ContextWrapper.startService()函数调用ContextImpl.startService()，ContextImpl.startService()最终又调用了ActivityManagerProxy.startServic()来实现启动服务的操作；
3. ActivityManagerService.startServic()里通过retrieveServiceLocked()解析在AndroidManifest.xml文件定义service标签时指定的android:process属性，然后通过ActivityManagerService.startProcessLocked创建一个新的进程；
4. 新进程执行Process.start导入ActivityThread，并执行ActivityThread.main()，然后调用ActivityThread.attach()；
5. ActivityThread.attach()把新进程的IApplicationThread实例通过Binder驱动程序传递给ActivityManagerService；
6. ActivityManagerService. scheduleCreateService调用ApplicationThread.scheduleCreateService回到新进程去启动service；
7. ActivityThread创建CreateServiceData放到消息队列中，并分发该消息；
8. ActivityThread获取到该消息后，调用handleCreateService()并通过ClassLoader加载srevice，然后调用service.onCreate()。
9. bindService
10. ContextWrapper.bindService()函数调用ContextImpl.bindService()，ContextImpl.bindService()最终又调用了ActivityManagerProxy.bindService()来实现启动服务的操作；
11. ActivityManagerProxy.bindService()通过token（activity在ActivityManagerService里面的令牌）获取代表activity的ActivityRecord；并通过retrieveServiceLocked函数解析在AndroidManifest.xml文件定义service标签，得到一个ServiceRecord（描述了Service对象）；最后将ActivityRecord和ServiceRecord封装成ConnectionRecord对象。
12. ActivityManagerService. scheduleCreateService调用ApplicationThread.scheduleCreateService回到主进程去启动service；
13. ActivityThread创建CreateServiceData放到消息队列中，并分发该消息；
14. ActivityThread获取到该消息后，调用handleCreateService()并通过ClassLoader加载srevice，然后调用service.onCreate()。
15. ActivityManagerService.requestServiceBindingsLocked调用 ApplicationThread.scheduleBindService()分发BIND\_SERVICE消息；调用ActivityThread.handleBindService()获取到Service，并调用其onBind()方法；调用ActivityManagerService.publishService()，进入到LoadedApk. ServiceDispatcher.InnerConnection.connected()转发给ServiceDispatcher. connected()，最后执行activity.onBind()入参ServiceConnection的onServiceConnected()方法。
16. startService和bindService的区别