https://blog.csdn.net/s332755645/article/details/73992860

**jcl-over-slf4j slf4j-log4j12等log工具作用**

2017年06月30日 14:15:04 [会走路的300快](https://me.csdn.net/s332755645) 阅读数 2316

java 界里有许多实现日志功能的工具，最早得到广泛使用的是 log4j，许多应用程序的日志部分都交给了 log4j，不过作为组件开发者，他们希望自己的组件不要紧紧依赖某一个工具，毕竟在同一个时候还有很多其他很多日志工具，假如一个应用程序用到了两个组件，恰好两个组件使用不同的日志工具，那么应用程序就会有两份日志输出了。

为了解决这个问题，Apache Commons Logging （之前叫 Jakarta Commons Logging，JCL）粉墨登场，JCL 只提供 log 接口，具体的实现则在运行时动态寻找。这样一来组件开发者只需要针对 JCL 接口开发，而调用组件的应用程序则可以在运行时搭配自己喜好的日志实践工具。

所以即使到现在你仍会看到很多程序应用 JCL + log4j 这种搭配，不过当程序规模越来越庞大时，JCL的动态绑定并不是总能成功，具体原因大家可以 Google 一下，这里就不再赘述了。解决方法之一就是在程序部署时静态绑定指定的日志工具，这就是 SLF4J 产生的原因。

跟 JCL 一样，SLF4J 也是只提供 log 接口，具体的实现是在打包应用程序时所放入的绑定器（名字为 slf4j-XXX-version.jar）来决定，XXX 可以是 log4j12, jdk14, jcl, nop 等，他们实现了跟具体日志工具（比如 log4j）的绑定及代理工作。举个例子：如果一个程序希望用 log4j 日志工具，那么程序只需针对 slf4j-api 接口编程，然后在打包时再放入 slf4j-log4j12-version.jar 和 log4j.jar 就可以了。

现在还有一个问题，假如你正在开发应用程序所调用的组件当中已经使用了 JCL 的，还有一些组建可能直接调用了 java.util.logging，这时你需要一个桥接器（名字为 XXX-over-slf4j.jar）把他们的日志输出重定向到 SLF4J，所谓的桥接器就是一个假的日志实现工具，比如当你把 jcl-over-slf4j.jar 放到 CLASS\_PATH 时，即使某个组件原本是通过 JCL 输出日志的，现在却会被 jcl-over-slf4j “骗到”SLF4J 里，然后 SLF4J 又会根据绑定器把日志交给具体的日志实现工具。过程如下

Component   
|   
| log to Apache Commons Logging   
V   
jcl-over-slf4j.jar — (redirect) —> SLF4j —> slf4j-log4j12-version.jar —> log4j.jar —> 输出日志

看到上面的流程图可能会发现一个有趣的问题，**假如在 CLASS\_PATH 里同时放置 log4j-over-slf4j.jar 和 slf4j-log4j12-version.jar 会发生什么情况呢？没错，日志会被踢来踢去，最终进入死循环。**

**总结：**

SLF4J，即简单日志门面（Simple Logging Facade for Java），不是具体的日志解决方案，它只服务于各种各样的日志系统。

slf4j-log4j12：如果仅仅引入slf4j，日志将不会生效。我们需要指定日志的具体实现类，slf4j-xxxx.jar,如slf4j-log4j12,将日志指定为log4j输出。

jcl-over-slf4j：把jcl实现的日志输出重定向到 SLF4J。