转载 java日志组件介绍（common-logging，log4j，slf4j，logback）

2012-12-10 00:55:19 yycdaizi 阅读数 40052 更多

common-logging

common-logging是apache提供的一个通用的日志接口。用户可以自由选择第三方的日志组件作为具体实现，像log4j，或者jdk自带的logging， common-logging会通过动态查
程序运行时自动找出真正使用的日志库。当然，common-logging内部有一个Simple logger的简单实现，但是功能很弱。所以使用common-logging，通常都是配合着log4j来使
好处就是，代码依赖是common-logging而非log4j， 避免了和具体的日志方案直接耦合，在有必要时，可以更改日志实现的第三方库。

使用common-logging的常见代码：

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class A {
        private static Log logger = LogFactory.getLog(this.getClass());
}
```

动态查找原理：Log 是一个接口声明。LogFactory 的内部会去装载具体的日志系统，并获得实现该Log 接口的实现类。LogFactory 内部装载日志系统的流程如下：

1. 首先，寻找org.apache.commons.logging.LogFactory 属性配置。
2. 否则，利用JDK1.3 开始提供的service 发现机制，会扫描classpah 下的META-INF/services/org.apache.commons.logging.LogFactory文件，若找到则装载里面的面的配置。
3. 否则，从Classpath 里寻找commons-logging.properties ，找到则根据里面的配置加载。
4. 否则，使用默认的配置：如果能找到Log4j 则默认使用log4j 实现，如果没有则使用JDK14Logger 实现，再没有则使用commons-logging 内部提供的SimpleLog 实

从上述加载流程来看，只要引入了log4j 并在classpath 配置了log4j.xml ，则commons-logging 就会使log4j 使用正常，而代码里不需要依赖任何log4j 的代码。

slf4j

slf4j全称为Simple Logging Facade for JAVA，java简单日志门面。类似于Apache Common-Logging，是对不同日志框架提供的一个门面封装，可以在部署的时候不修改
接入一种日志实现方案。但是，他在编译时静态绑定真正的Log库。使用SLF4J时，如果你需要使用某一种日志实现，那么你必须选择正确的SLF4J的jar包的集合（各种桥接包
使用slf4j的常见代码：

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class A {
        private static Log logger = LogFactory.getLog(this.getClass());
}
```

slf4j静态绑定原理：SLF4J 会在编译时会绑定import org.slf4j.impl.StaticLoggerBinder; 该类里面实现对具体日志方案的绑定接入。任何一种基于slf4j 的实现有
如：org.slf4j.slf4j-log4j12-1.5.6: 提供对 log4j 的一种适配实现。**注意**：如果有任意两个实现slf4j 的包同时出现，那么就可能出现问题。

slf4j 与 common-logging 比较

common-logging通过动态查找的机制，在程序运行时自动找出真正使用的日志库。由于它使用了ClassLoader寻找和载入底层的日志库， 导致了象OSGI这样的框架无法
的不同的插件使用自己的ClassLoader。 OSGI的这种机制保证了插件互相独立，然而却使Apache Common-Logging无法工作。

slf4j在编译时静态绑定真正的Log库,因此可以再OSGI中使用。另外，SLF4J 支持参数化的log字符串，避免了之前为了减少字符串拼接的性能损耗而不得不写的
if(logger.isDebugEnable())，现在你可以直接写：logger.debug(“current user is: {}”, user)。拼装消息被推迟到了它能够确定是不是要显示这条消息的时候
价并没有幸免。

Log4j

Apache的一个开放源代码项目，通过使用Log4j，我们可以控制日志信息输送的目的地是控制台、文件、GUI组件、甚至是套接口服务 器、NT的事件记录器、UNIX Sys.og守护
也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，用户能够更加细致地控制日志的生成过程。这些可以通过一个 配置文件来灵活地进行配置，而不需要
码。

LogBack

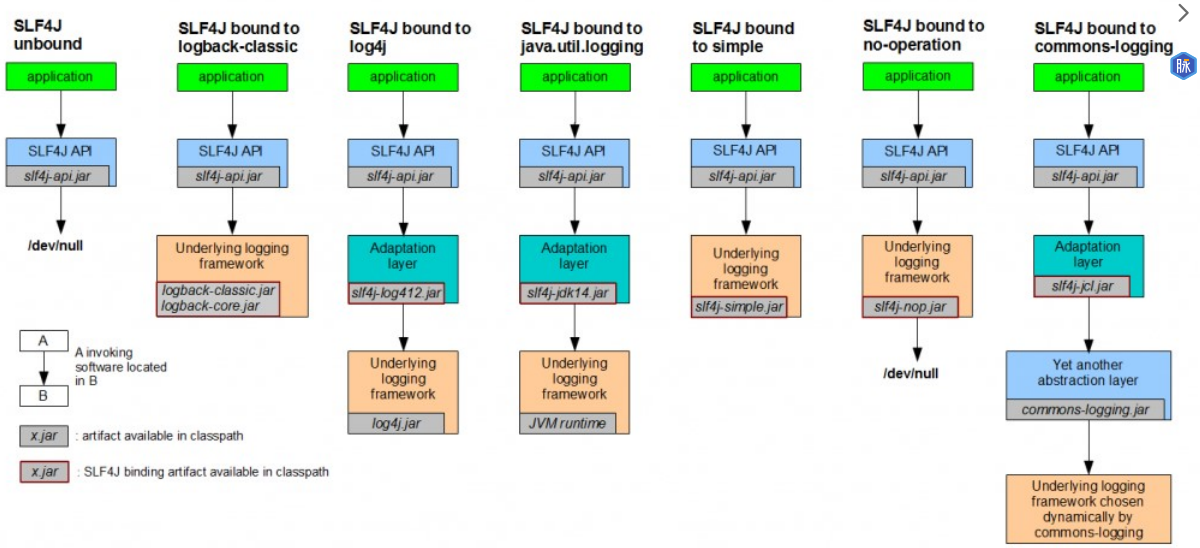
Logback是由log4j创始人设计的又一个开源日记组件。logback当前分成三个模块：logback-core,logback- classic和logback-access。logback-core是其它两个模块的基础模块。logback-classic是log4j的一个 改良版本。此外logback-classic完整实现SLF4J API使你可以很方便地更换成其它日记系统如log4j或JDK14 Logging。logback-access访问容器集成提供通过Http来访问日记的功能。

Log4j 与 LogBack 比较

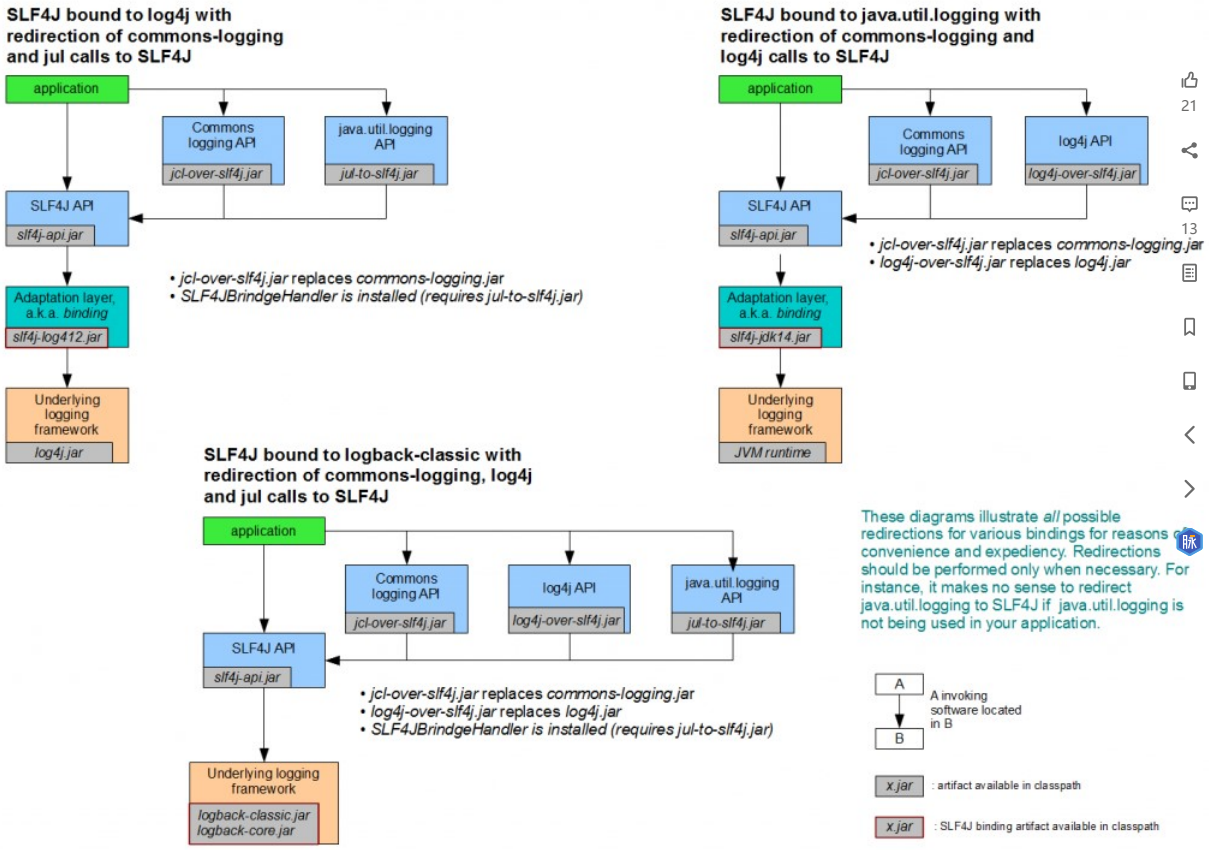
LogBack作为一个通用可靠、快速灵活的日志框架，将作为Log4j的替代和SLF4J组成新的日志系统的完整实现。LOGBack声称具有极佳的性能，“ 某些关键操作，比如判定是否记录一条日志语句的操作，其性能得到了显著的提高。这个操作在LogBack中需要3纳秒，而在Log4J中则需要30纳秒。 LogBack创建记录器（logger）的速度也更快：13微秒，而在Log4J需要2234微秒。更重要的是，它获取已存在的记录器只需94纳秒，而 Log4J需要2234纳秒，时间减少到了1/23。跟JUL相比的性能提高也是显著的”。 另外，LOGBack的所有文档是全面的、而且都是免费的。不象Log4J那样只提供部分免费文档而需要用户去购买付费文档。

slf4j与其他各种日志组件的桥接

应用代码中使用slf4j接口，接入具体实现的方法



应用代码中使用别的日志接口，转成slf4j的方法



日志组件相关历史

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, jc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 Xslf4j.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，最终进入死循环。

所以使用 SLF4J 的比较典型搭配就是把 slf4j-api、JCL 桥接器、java.util.logging（JUL）桥接器、log4j 绑定器、log4j 这5个 jar 放置在 CLASS_PATH 里。

不过并不是所有APP容器都是使用 log4j 的，比如 Google AppEngine 它使用的是 java.util.logging（JUL），这时应用 SLF4J 的搭配就变成 slf4j-api、JCL桥接器、loJUL绑定器这4个 jar 放置在 WEB-INF/lib 里。