<http://sagewsg.iteye.com/blog/1486947>

**一、     概念**

**Commons-logging**: apache最早提供的日志的门面接口。避免和具体的日志方案直接耦合。类似于JDBC 的api 接口，具体的的JDBC driver 实现由各数据库提供商实现。通过统一接口解耦，不过其内部也实现了一些简单日志方案。

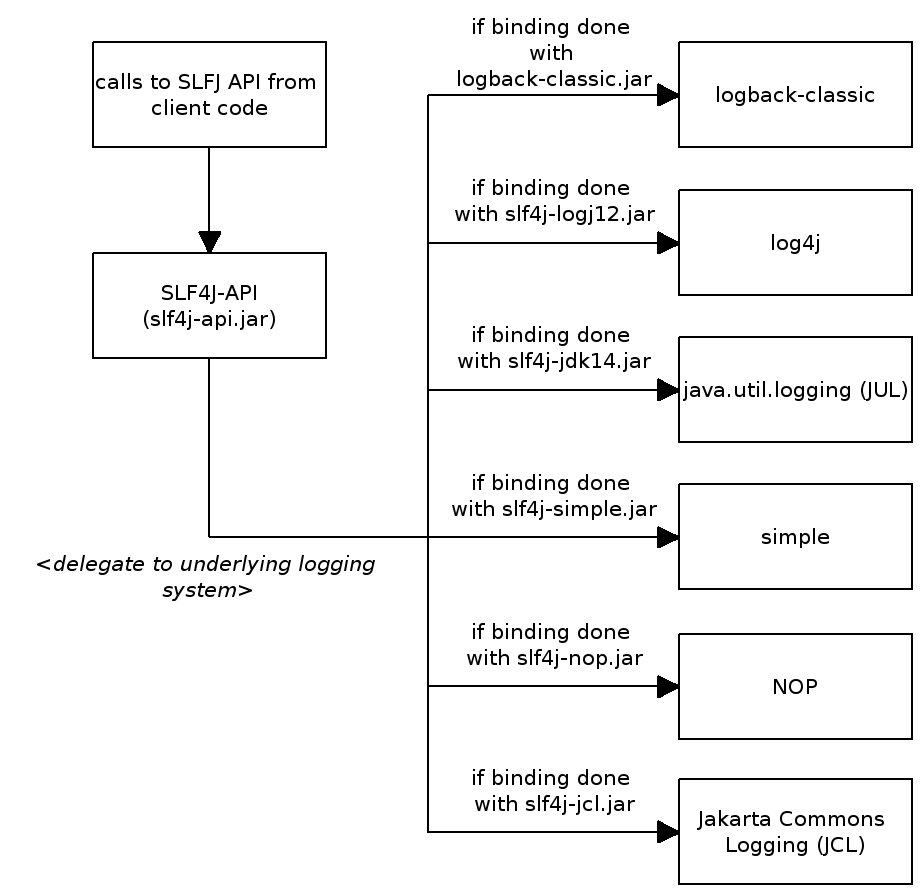
**Log4j**: 经典的一种日志解决方案。内部把日志系统抽象封装成Logger 、appender 、pattern 等实现。我们可以通过配置文件轻松的实现日志系统的管理和多样化配置。

**Slf4j**: 全称为Simple Logging Facade for JAVA：java简单日志门面。 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任何配置即可接入一种日志实现方案。和commons-loging 应该有一样的初衷。个人感觉设从计上更好一些，没有commons 那么多潜规则。同时有两个额外特点：

1. 能支持多个参数，并通过{} 占位符进行替换，避免老写logger.isXXXEnabled 这种无奈的判断，带来性能提升见：http://www.slf4j.org/faq.html#logging\_performance 。

2.OSGI 机制更好兼容支持

一图胜千言，官网上的一个图：



从上图可以发现，选择还是很多的。

**Logback**: 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。官网上称具有极佳的性能，在关键路径上执行速度是log4j 的10 倍，且内存消耗更少。具体优势见：

http://logback.qos.ch/reasonsToSwitch.html

**二、     常见日志方案和注意事项**

**1.Commons-logging+log4j**: 经典的一个日志实现方案。出现在各种框架里。如spring 、webx 、ibatis 等等。直接使用log4j 即可满足我们的日志方案。但是一般为了避免直接依赖具体的日志实现，一般都是结合commons-logging 来实现。常见代码如下：

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

private static Log logger = LogFactory.getLog(CommonsLoggingTest.class);

代码上，没有依赖任何的log4j 内部的类。那么log4j 是如何被装载的？

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。而内部有一个Log4JLogger 实现类对Log 接口同时内部提供了对log4j logger 的代理。LogFactory 内部装载日志系统流程：

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配置，使用里面的配置。

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 实现。

从上述加载流程来看，如果没有做任何配置，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。

**2.Commons-logging+log4j+slf4j**

如果在原有commons-logging 系统里，如果要迁移到slf4j, 使用slf4j 替换commons-logging ，也是可以做到的。原理使用到了上述commons-logging 加载的第二点。需要引入Org.slf4j.jcl-over-slf4j-1.5.6.jar 。这个jar 包提供了一个桥接，让底层实现是基于slf4j 。原理是在该jar 包里存放了配置META-INF/services/org.apache.commons.logging.LogFactory =org.apache.commons.logging.impl.SLF4JLogFactory ，而commons-logging 在初始化的时候会找到这个serviceId ，并把它作为LogFactory 。

完成桥接后，那么那么简单日志门面SLF4J 内部又是如何来装载合适的log 呢？

原理是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 的实现都要有一个这个类。如：

org.slf4j.slf4j-log4j12-1.5.6: 提供对 log4j 的一种适配实现。

Org.slf4j.slf4j-simple-1.5.6: 是一种 simple 实现，会将 log 直接打到控制台。

……

那么这个地方就要注意了：如果有任意两个实现slf4j 的包同时出现，那就有可能酿就悲剧，你可能会发现日志不见了、或都打到控制台了。原因是这两个jar 包里都有各自的org.slf4j.impl.StaticLoggerBinder ，编译时候绑定的是哪个是不确定的。这个地方要特别注意！！出现过几次因为这个导致日志错乱的问题。

**3.Slf4j+logback**

Slf4j 和log4j 作者都是同一个人。

Logback 号称在性能各方面有很多优势，也很诱人。

直接使用SLf4j 也很简单：

**Java代码  [收藏代码](javascript:void())**

1. **import** org.slf4j.Logger;
2. **import** org.slf4j.LoggerFactory;
4. **public** **class** HelloWorld {
5. **public** **static** **void** main(String[] args) {
6. Logger logger = LoggerFactory.getLogger(HelloWorld.**class**);
7. logger.info("Hello World");
8. }
9. }

代码里也看不到任何具体日志实现方案的痕迹。

Logback 没用过，看到过一些诱人介绍。具体大家可以去研究。logback 。

**注意事项**

使用日志配置的时候一定要明白需求，同时避免冲突。

如使用SLF4j 的时候为了避免冲突，一定要保障只有一种实现类jar 包在里面。

当遇到日志错乱等问题时，可以从这几个方面来排查

二.slf4j

下边一段程序是经典的使用slf4j的方法.

**Java代码  [收藏代码](javascript:void())**

1. **import** org.slf4j.Logger;
2. **import** org.slf4j.LoggerFactory;
3. **public** **class** Wombat {
4. **final** Logger logger = LoggerFactory.getLogger(Wombat.**class**);
5. Integer t;
6. Integer oldT;
7. **public** **void** setTemperature(Integer temperature) {
8. oldT = t;
9. t = temperature;
10. logger.error("Temperature set to {}. Old temperature was {}.", t, oldT);
11. **if** (temperature.intValue() > 50) {
12. logger.info("Temperature has risen above 50 degrees.");
13. }
14. }
15. **public** **static** **void** main(String[] args) {
16. Wombat wombat = **new** Wombat();
17. wombat.setTemperature(1);
18. wombat.setTemperature(55);
19. }
20. }

下边介绍一下运行上边程序的过程。  
1,编译上边的程序，需要classpath中加入slf4j-api-1.4.1.jar文件  
2,运行时，需要classpath中加上slf4j-simple-1.4.1.jar  
运行得到结果:  
----------------------------

**输出代码  [收藏代码](javascript:void())**

1. 0 [main] ERROR Wombat - Temperature set to 1. Old temperature was null.
2. 0 [main] ERROR Wombat - Temperature set to 55. Old temperature was 1.
3. 0 [main] INFO Wombat - Temperature has risen above 50 degrees.

这个是simple log风格，

3,切换：如果想切换到jdk14的log的风格，只需要把slf4j-simple-1.4.1.jar  
从classpath中移除，同时classpath中加入slj4j-jdk14-1.4.1.jar  
这时的运行结果:  
---------------------------------------------------

**输出代码  [收藏代码](javascript:void())**

1. 2007-7-9 10:40:15 Wombat setTemperature
2. 严重: Temperature set to 1. Old temperature was null.
3. 2007-7-9 10:40:16 Wombat setTemperature
4. 严重: Temperature set to 55. Old temperature was 1.
5. 2007-7-9 10:40:16 Wombat setTemperature
6. 信息: Temperature has risen above 50 degrees.

已经变成jdk14的log风格了。

4,再次切换到log4j  
同样移除slj4j-jdk14-1.4.1.jar，加入slf4j-log4j12-1.4.1.jar，同时加入log4j-1.2.x.jar  
加入log4j.properties。得到显示结果:  
---------------------------------------

**输出代码  [收藏代码](javascript:void())**

1. 10:42:27,328 ERROR Wombat: Temperature set to 1. Old temperature was null.
2. 10:42:27,328 ERROR Wombat: Temperature set to 55. Old temperature was 1.
3. 10:42:27,328  INFO Wombat: Temperature has risen above 50 degrees.

在不同的风格中切换只需要在部署期切换类库就可以了，和开发时无关。

三.log4j 在spring等项目中web.xml的配置

**Xml代码  [收藏代码](javascript:void())**

1. **<context-param>**
2. **<param-name>**webAppRootKey**</param-name>**
3. **<param-value>**spring\_mvc\_xml.root**</param-value>**
4. **</context-param>**
5. <!-- 开一条watchdog线程每3秒扫描一下配置文件的变化; -->
6. **<context-param>**
7. **<param-name>**log4jRefreshInterval**</param-name>**
8. **<param-value>**3000**</param-value>**
9. **</context-param>**
10. **<context-param>**
11. **<param-name>**log4jConfigLocation**</param-name>**
12. **<param-value>**/WEB-INF/classes/log4j.properties**</param-value>**
13. **</context-param>**
14. **<listener>**
15. **<listener-class>**org.springframework.web.util.Log4jConfigListener**</listener-class>**
16. **</listener>**
17. <!--这里配置了一个listener，也可以配置servlet:-->
18. <!--以下为非spring的servlet配置-->
19. **<servlet>**
20. **<servlet-name>**log4j**</servlet-name>**
21. **<servlet-class>**
22. org.springframework.web.util.Log4jConfigServlet
23. **</servlet-class>**
24. **<load-on-startup>**1**</load-on-startup>**
25. **</servlet>**

转自：<http://www.iteye.com/topic/934593>

<http://www.blogjava.net/dreamstone/archive/2007/07/09/128993.html>