slf4j acts as a simple facade for various logging frameworks (jul, log4j, jcl, logback) and allows you to plug in the desired logging framework at deployment time.

<http://java.dzone.com/articles/adding-slf4j-your-maven>

# Adding SLF4J to your Maven Project

09.27.2011

 | 38685 views |

[Tweet](http://twitter.com/share)

[inShare](javascript:void(0);)1

* [Tweet](http://twitter.com/share)
* 1  
  [inShare](javascript:void(0);)
* [submit to reddit](http://www.reddit.com/r/programming/submit)

In yesterday’s blog, I documented the various bits of the Maven POM file created for a ‘Spring MVC Project’ using one of Spring’s project templates. In that POM file you may have noticed a reference to an slf4j-api artefact and comment to the effect that the Guy’s at Spring were excluding “Commons Logging in favor of SLF4j”. This blog takes a look at SLF4j, or to give it its full title: “[Simple Logging Facade for Java](http://www.slf4j.org/index.html)” and demonstrates how to add it to a project.  
  
SLF4j is not a logger in itself, it’s a facade or wrapper that delegates the actual business of logging to one of the more well known logger implementations, which is usually Log4J. The idea behind it is that’s its a replacement for that other well known logging facade: Commons Logging. The reason that it has been written as an alternative to Commons Logging, is that Commons Logging loads your logging library by using some whizzy Java ClassLoader techniques. Unfortunately, this has gained it a reputation for being somewhat buggy (although it has to be said I’ve never come across any problems with Commons Logging).  
  
SLF4j, on the other hand, does things more simply; there’s no whizzy ClassLoader, you simply specify the slf4j API library, the one you use in your application, plus another SLF4j library that binds the first library to the logger implementation you’re using such as Log4J. The shortlist of binding libraries that SLF4j supports can be found [here](http://www.slf4j.org/manual.html#binding).  
  
To add SLF4j to your project, the first thing to is to add in the SLF4j API

[view source](http://java.dzone.com/articles/adding-slf4j-your-maven#viewSource)

[print](http://java.dzone.com/articles/adding-slf4j-your-maven#printSource)[?](http://java.dzone.com/articles/adding-slf4j-your-maven#about)

1.<properties>

2.<slf4jVersion>1.6.1</slf4jVersion>

3.</properties>

4.<dependency>

5.<groupId>org.slf4j</groupId>

6.<artifactId>slf4j-api</artifactId>

7.<version>${slf4jVersion}</version>

8.</dependency>

...which is pretty simple. The thing to note is that this must be in compile scope. The next step is to choose a binding library; one of the following:

[view source](http://java.dzone.com/articles/adding-slf4j-your-maven#viewSource)

[print](http://java.dzone.com/articles/adding-slf4j-your-maven#printSource)[?](http://java.dzone.com/articles/adding-slf4j-your-maven#about)

01.<!-- CHOOSE BETWEEN ONE OF THESE DIFFERENT BINDINGS -->

02.<!-- Binding for NOP, silently discarding all logging. -->

03.<dependency>

04.<groupId>org.slf4j</groupId>

05.<artifactId>slf4j-nop</artifactId>

06.<version>${slf4jVersion}</version>

07.</dependency>

08.<!-- Binding for System.out -->

09.<dependency>

10.<groupId>org.slf4j</groupId>

11.<artifactId>slf4j-simple</artifactId>

12.<version>${slf4jVersion}</version>

13.</dependency>

14.<!--Binding for log4j version 1.2.x You also need to

15.place log4j.jar on your class path. -->

16.<dependency>

17.<groupId>org.slf4j</groupId>

18.<artifactId>slf4j-log4j12</artifactId>

19.<version>${slf4jVersion}</version>

20.</dependency>

21.<dependency>

22.<groupId>log4j</groupId>

23.<artifactId>log4j</artifactId>

24.<version>1.2.16</version>

25.<scope>runtime</scope>

26.</dependency>

27.<!--Binding for commons logging over slf4j -->

28.<dependency>

29.<groupId>org.slf4j</groupId>

30.<artifactId>jcl-over-slf4j</artifactId>

31.<version>${slf4jVersion}</version>

32.<scope>runtime</scope>

33.</dependency>

...and note that the scope here can be runtime.  
  
If you add more than one binding JAR to your project config, then you'll get the following error message:

[view source](http://java.dzone.com/articles/adding-slf4j-your-maven#viewSource)

[print](http://java.dzone.com/articles/adding-slf4j-your-maven#printSource)[?](http://java.dzone.com/articles/adding-slf4j-your-maven#about)

1.SLF4J: Class path contains multiple SLF4J bindings.

2.SLF4J: Found binding in [jar:file:/Users/Roger/.m2/repository/org/slf4j/slf4j-nop/1.6.1/slf4j-nop-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]

3.SLF4J: Found binding in [jar:file:/Users/Roger/.m2/repository/org/slf4j/slf4j-log4j12/1.6.1/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]

4.SLF4J: See <http://www.slf4j.org/codes.html>#multiple\_bindings for an explanation.

...and alternatively, if you forget to add any bindings, the following message is written to Standard Err and all log messages are sent to nop (no/null output):

[view source](http://java.dzone.com/articles/adding-slf4j-your-maven#viewSource)

[print](http://java.dzone.com/articles/adding-slf4j-your-maven#printSource)[?](http://java.dzone.com/articles/adding-slf4j-your-maven#about)

1.SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

2.SLF4J: Defaulting to no-operation (NOP) logger implementation

3.SLF4J: See <http://www.slf4j.org/codes.html>#StaticLoggerBinder for further details.

The following code demonstrates how to use SLF4j in an application, as you can see the fatal logging level has not been implemented.

[view source](http://java.dzone.com/articles/adding-slf4j-your-maven#viewSource)

[print](http://java.dzone.com/articles/adding-slf4j-your-maven#printSource)[?](http://java.dzone.com/articles/adding-slf4j-your-maven#about)

01.public static void main(String[] args) {

02.

03.Logger logger = LoggerFactory.getLogger(Slf4jHelloWorld.class);

04.

05.logger.trace("Hello World");

06.logger.debug("Hello World");

07.logger.info("Hello World");

08.logger.warn("Hello World");

09.logger.error("Hello World");

10.}

SLF4j is well documented and you can find all the [manuals](http://www.slf4j.org/docs.html) and what-not on their site.  
  
Finally, at the time of writing, the lastest version of SLF4j is 1.6.2; however, this only seems to be available from the SLF4j website. The lastest version available from Maven Central is 1.6.1.

DEBUG LEVELS

OFF

FATAL

ERROR

WARN

INFO

DEBUG

TRACE

ALL

IF PUT DEBUG, it wont print trace and all. If info then it won’t debug ,trace and all