SLF4J

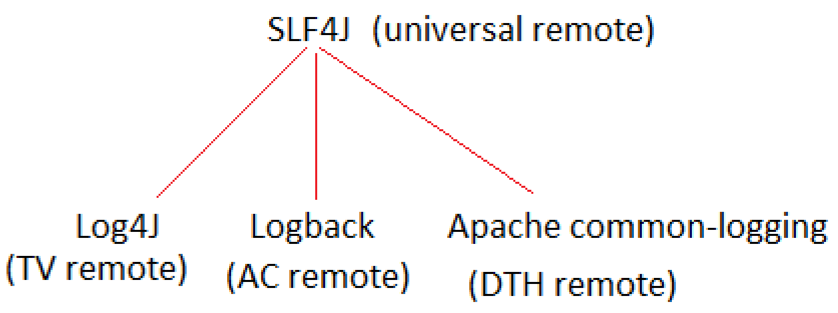
Simple Logging Facade for Java

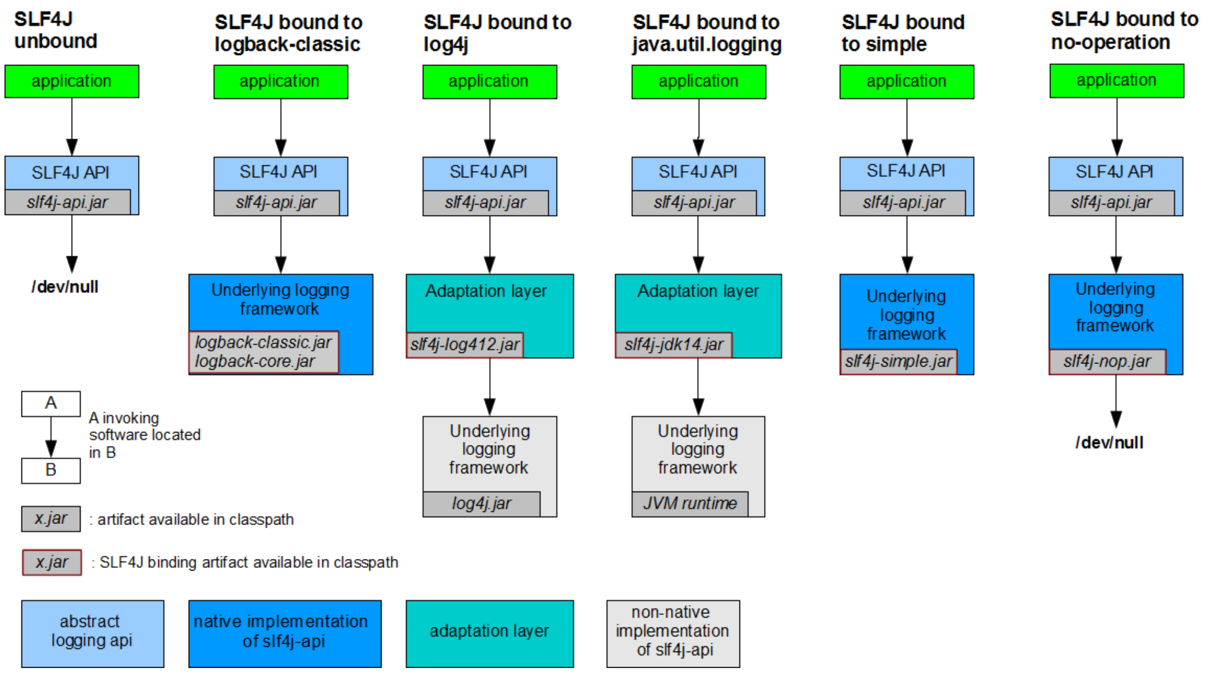
INDEX

SLF4J

SLF4J

- SLF$J stands for Simple Logging Facade for Java.
- To work with different logging tools, we need to use different APIs. So, moving from one logging tool to another logging tool becomes very difficult.
- To overcome this problem SLF4J is given, which provides abstraction multiple logging tools or API and provides unified API to work any logging tool/ API.




SLF4J unbound
SLF4J bound to logback-classic
SLF4J bound to log4j
SLF4J bound to java.util.logging
SLF4J bound to simple
SLF4J bound to no-operation
application
SLF4J API
slf4j-api.jar
/dev/null
Underlying logging framework
logback-classic.jar
logback-core.jar
Adaptation layer
slf4j-log412.jar
slf4j-jdk14.jar
slf4j-simple.jar
slf4j-nop.jar
Underlying logging framework
log4j.jar
JVM runtime
A
B
A invoking software located in B
x.jar : artifact available in classpath
x.jar : SLF4J binding artifact available in classpath
abstract logging api
native implementation of slf4j-api
adaptation layer
non-native implementation of slf4j-api

Binding with a logging framework at deployment time

- SLF4J supports various logging frameworks. The SLF4J distribution ships with several jar files referred to as "SLF4J bindings", with each binding corresponding to a supported framework.

- slf4j-log4j12-${latest.stable.version}.jar
  Binding for log4j version 1.2, a widely used logging framework. You also

need to place log4j.jar on pour class path.

- slf4j-jdk14-${latest.stable.version}.jar
  Binding for java.util.logging, also referred to as JDK 1.4 logging.

- slf4j-nop-${latest.stable.version}.jar
  Binding for NOP, silently discarding all logging.

- slf4j-simple-${latest.stable.version}.jar
  Binding for Simple implementation, which outputs all events to System.err. Only messages of level INFO and higher are printed. This binding may be useful in the context of small applications.

- slf4j-jcl-${latest.stable.version}.jar
  Binding for Jakarta Commons Logging. This binding will delegate all SLF4J logging to JCL.

- logback-classic-${logback. version}.jar (requires logback-core-${logback. version}.jar)
  There are also SLF4J bindings external to the SLF4J project, e.g., logback which implements SLF4J natively. Logback's ch.qos.logbacl.classic.Logger class is a direct implementation of SLf4J's org.slf4j.Logger interface.

The logger levels of SLF4J are:

- Debug < info < trace < warn < error (No fatal here)
- For user activity related event handling-based code execution will be logged with the support of "trace" log message (Auditing activities).
- Button is clicked actionPerformed(-) method executed this can be logged through "trace" level.

Procedure to add SLF4J with Log4j 1.x support to Java application for logging

Step 1: Add the following jar files to the CLASSPATH or BUILDPATH

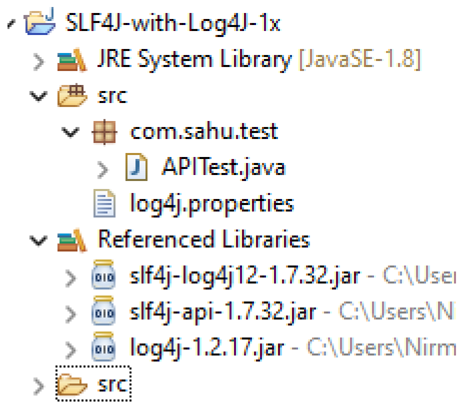
- slf4j-api-<version>.jar [Click here]
- slf4j-log4j12-<version>.jar [Click here]
- log4j-<version>.jar [Click here]

Step 2: Place log4j.properties file in "src" folder.
Step 3: Develop any java code having SLF4J based log messages.

Note:

- Here SLFJ4 generated the log messages by using Log4J setup internally based on the instructions collected from the log4j.properties file.
- Spring boot internally use SLF4J with Log4jJ to generate log messages, we can control these log messages through application.properties file.

Directory Structure of SLF4J-with-Log4J-1x:



```
SLF4J-with-Log4J-1x
  > JRE System Library [JavaSE-1.8]
  v src
    v com.sahu.test
      > APITest.java
      log4j.properties
  v Referenced Libraries
    > slf4j-log4j12-1.7.32.jar - C:\User
    > slf4j-api-1.7.32.jar - C:\Users\N
    > log4j-1.2.17.jar - C:\Users\Nirm
  > src
```

- develop the above project and package and classes, properties.
- Add the give jar to their build path and place the following code with their respective files.

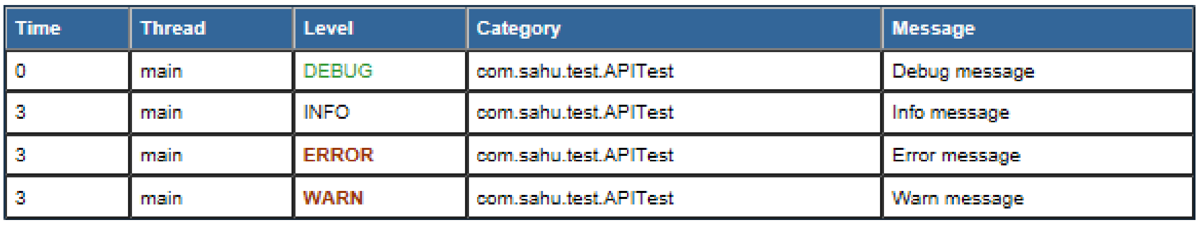
Log4j.properties

```
#For HTMLLayout and FileAppender

#Specify logger level to retrieve the log messages
log4j.rootLogger=DEBUG, R

#Appender SetUp
#Specify the FileAppender
log4j.appender.R=org.apache.log4j.FileAppender
#Specify the File Name and location
log4j.appender.R.File=info.html
#Disabling append mode on file
log4j.appender.R.append=false

#Layout SetUp
#Specify the HTMLLayout
log4j.appender.R.layout=org.apache.log4j.HTMLLayout
```

APITest.java

```
package com.sahu.test;

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

public class APITest {

	private static Logger logger = LoggerFactory.getLogger(APITest.class);

	public static void main(String[] args) {
		logger.debug("Debug message");
		logger.info("Info message");
		logger.trace("Trace message");
		logger.error("Error message");
		logger.warn("Warn message");
	}

}
```

Run the application then refresh the project, you will get the info.html



Time	Thread	Level	Category	Message
0	main	DEBUG	com.sahu.test.APITest	Debug message
3	main	INFO	com.sahu.test.APITest	Info message
3	main	ERROR	com.sahu.test.APITest	Error message
3	main	WARN	com.sahu.test.APITest	Warn message

-- The END --