SLF4J

Getting Started

SLF4J(Simple Logging Facade For Java) offers the abstract layer to set logs free from the sepcific logging service implementor. Use of SLF4J in combination with Jakarta Commons Logging(JCL), Log4j and Logback is preferred. Refer to the following example for how to use SLF4J:

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

class Slf4JLoggerTest {

    // Generating logger objects using SLF4J
    private static final Logger LOGGER = LoggerFactory.getLogger(Slf4JLoggerTest.class);

    // Parameterized logging - String Type
    String message = "Hello, eGovFrame 3.0";
    String message2 = "Welcome to eGovFrame 3.0";

    LOGGER.debug("SLF4J Logger - {}", message); // Output - SLF4J Logger - Hello, eGovFrame 3.0
    LOGGER.debug("SLF4J Logger - {} and {}", message, message2); // Output - SLF4J Logger - Hello, eGovFrame
3.0 and Welcome to eGovFrame 3.0

    // Parameterized logging - Object type
    Object[] args = new Object[3]; args[0]
    = "1";
    args[1] = Integer.valueOf("2");
    args[2] = new Date().toString();

    LOGGER.debug("SLF4J Logger - {}, {}, {}", args); // Output - SLF4J Logger - 1, 2, Fri Mar 23 11:08:28 KST
2014
}
```

1. Basic SLF4J configuration

1) In order to use SLF4J API, you need to add **slf4j-api.jar** as follows:

```
<!-- SLF4J -->
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>x.x.x</version>
</dependency>
```

2) You then need to remove the Logging Framework of Spring, **commons-logging.jar**, in prevention of crash. Make sure to add the SLF4j JCL Bridge, **jcl-over-slf4j.jar**, for you to convert Commons Logging API as required.

```
<!-- Exclude Commons Logging in favor of SLF4J -->
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>${spring.maven.artifact.version}</version>
   <exclusions>
      <exclusion>
         <groupId>commons-logging</groupId>
         <artifactId>commons-logging</artifactId>
      </exclusion>
   </exclusions>
</dependency>
```

```
<!-- SLF4J JCL Bridge -->
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>jcl-over-slf4j</artifactId>
   <version>x.x.x</version>
</dependency>
```

2. Configuring logging implementors

1) Add **SLF4J Binding jar and Implementation jar** to make logging implementors available when SLF4J is complied.

Logging Implementor	SLF4J Binding jar
Log4j 2	log4j-slf4j-impl.jar
Log4j 1.2	slf4j-log4j12.jar
JDK 1.x Logging	slf4j-jdk14.jar
NOP	slf4j-nop.jar
JCL	slf4j-jcl.jar
Logback	logback-classic.jar, logback-core.jar

- When using Log4j 1.2 implementor

```
<!--    SLF4J Log4j1.2 Binding -->
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-log4j12</artifactId>
   <version>x.x.x</version>
</dependency>

<!-- Log4j 1.2 -->
<dependency>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
   <version>1.2</version>
</dependency>
```

- When using Log4j 2 implementor

```
<!-- Log4j2 SLF4J Binding -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-slf4j-impl</artifactId>
   <version>x.x.x</version>
</dependency>

<!-- Log4j 2 -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-api</artifactId>
   <version>x.x.x</version>
</dependency>
```

3. Generating SLF4J logger objects ahd using methods

1) Generating Logger objects

```
private static final Logger LOGGER = LoggerFactory.getLogger(Slf4JLoggerTest.class);
```

2) Calling logging methods

```
// Parameterized Logging using {}-placeholder
String message = "Hello, eGovFrame 3.0";

LOGGER.debug("SLF4J Logger - {}", message);
```

Migration to SLF4J from Legacy APIs

You need to secure a bridge jar that links SLF4J and legacy API to use both Legacy API and SLF4J.
Se the following example for how to do so, assuming Log4j 1.x and JCL legacy:

1. Replacing logging implementor jar into SLF4J Bridge jar

If you replace logging implementor jar into SLF4J Bridge jar, you hand SLF4J off to the right to logging control of the implementor.

1) Replacing log4j.jar into log4j-over-slf4j.jar while maintaining Log4j 1.x

```
<!-- Log4j 1.x -->
<!--
<dependency>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
   <version>x.x.x</version>
</dependency>
-->

<!-- SLF4J Log4j 1.x Bridge -->
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>log4j-over-slf4j</artifactId>
   <version>x.x.x</version>
</dependency>
```

(Caution) Concurrent use of log4j-over-slf4j.jar and slf4j-log4j12.jar(SLF4j Binding) not supported

2) Replacing commons-logging.jar into jcl-over-slf4j.jar while maintaining JCL(Jakarta Commons Logging)

```
<!-- Commons Logging -->
<!--
<dependency>
   <groupId>commons-logging</groupId>
   <artifactId>commons-logging</artifactId>
   <version>1.1.1</version>
</dependency>
-->

<!-- SLF4j JCL Bridge -->
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>jcl-over-slf4j</artifactId>
   <version>x.x.x</version>
</dependency>
```

2. Replacing environment configuration file into logback

SLF4J does not recognize the environment configuration file for log4j which must be converted into a logback. You can simply use
log4j properties file translator or refer to logback manual for how to make a conversion.

Refer to the following example for logback.xml:

```
<configuration>
   <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
      <!-- encoders are assigned the type
```

```
            ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="debug">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

References

- SLF4J
- Logback