



# sli4j

---

**sli4j**  
v. 1.0  
User Guide

---



## Table of Contents

---

1	<b>Table of Contents</b> .....	<b>i</b>
2	<b>Home</b> .....	<b>1</b>
3	<b>JULI</b> .....	<b>2</b>
4	<b>Apache Commons Logging</b> .....	<b>3</b>
5	<b>Apache log4j</b> .....	<b>4</b>
6	<b>Simple Logging Facade for Java</b> .....	<b>5</b>
7	<b>Extend sli4j</b> .....	<b>6</b>



# 1 Home

---

## 1.1 Introduction

The **sli4j**, acronymous of *Simple Logger Injector for Java*, is a small, light and fast logger Injector, built on top of *Google-Guice*, for the well known frameworks:

- Java Utils Logging;
- Apache [commons-logging](#);
- Apache [log4j](#);
- Simple Logging Facade for Java [slf4j](#), with [Logback](#) support.

The concept behind **sli4j** is that instead of creating Loggers by hand, users can let *Guice* creates and injects them automagically, for example instead writing:

```
import java.util.logging.Logger;
...
Logger logger = Logger.getLogger(this.getClass().getName());
...
```

users can easily code:

```
import java.util.logging.Logger;
...
Logger logger;
...
```

and **nothing** more! **No** setter methods are needed, **no** special annotations, just declare it and let *sli4j* doing the rest, `final` and already set Loggers will be skipped and *sli4j* won't try to override them at all.

## 1.2 Before Coding...

To set up your project, configure in your `pom.xml` the repository:

```
<repositories>
...
<repository>
  <id>sli4j-repository</id>
  <name>sli4j Repository for Maven</name>
  <url>http://sli4j.googlecode.com/svn/repo</url>
  <layout>default</layout>
</repository>
...
</repositories>
```

## 1.3 Acknowledgements

This work is dedicated to our city, L'Aquila, destroyed by a terrible earthquake the 6th April, 2009... That day more than 300 people were killed because buildings collapsed after a magnitudo 6.3 earthquake at 3:32 am.

We'll never forget that episode.

## 2 JULI

---

### 2.1 JULI - Java Utils Logging

Users that want to use the `java.util.logging` package and let *Guice* injects automatically `java.util.logging.Logger` instances, have to add the following dependency in the `pom.xml`:

```
<dependency>
  <groupId>com.google.code.sli4j</groupId>
  <artifactId>sli4j-juli</artifactId>
  <version>XX.XX</version>
  <scope>compile</scope>
</dependency>
```

then, when creating the `com.google.inject.Injector`, add the `com.google.code.sli4j.juli.JuliLoggingModule` module; please take note that users have to specify the classes `com.google.inject.matcher.Matcher` for whom the logging injection has to be applied:

```
import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.code.sli4j.juli.JuliLoggingModule;
import com.google.inject.matcher.Matchers;
...
Injector injector = Guice.createInjector(new JuliLoggingModule(Matchers.any()),
    ...
);
```

and the magic happens :)

## 3 Apache Commons Logging

---

### 3.1 Apache Commons Logging

Users that want to use the *Apache Commons Logging* package and let *Guice* injects automatically `org.apache.commons.logging.Log` instances, have to add the following dependency in the `pom.xml`:

```
<dependency>
  <groupId>com.google.code.sli4j</groupId>
  <artifactId>sli4j-acl</artifactId>
  <version>XX.XX</version>
  <scope>compile</scope>
</dependency>
```

then, when creating the `com.google.inject.Injector`, add the `com.google.code.sli4j.acl.ACLLoggingModule` module; please take note that users have to specify the classes `com.google.inject.matcher.Matcher` for whom the logging injection has to be applied:

```
import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.code.sli4j.acl.ACLLoggingModule;
import com.google.inject.matcher.Matchers;
...
Injector injector = Guice.createInjector(new ACLLoggingModule(Matchers.any()),
    ...
);
```

and the magic happens :)

## 4 Apache log4j

---

### 4.1 Apache log4j

Users that want to use the *Apache log4j* package and let *Guice* injects automatically `org.apache.log4j.Logger` instances, have to add the following dependency in the `pom.xml`:

```
<dependency>
  <groupId>com.google.code.sli4j</groupId>
  <artifactId>sli4j-log4j</artifactId>
  <version>XX.XX</version>
  <scope>compile</scope>
</dependency>
```

then, when creating the `com.google.inject.Injector`, add the `com.google.code.sli4j.log4j.Log4jLoggingModule` module; please take note that users have to specify the classes `com.google.inject.matcher.Matcher` for whom the logging injection has to be applied:

```
import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.code.sli4j.log4j.Log4jLoggingModule;
import com.google.inject.matcher.Matchers;
...
Injector injector = Guice.createInjector(new Log4jLoggingModule(Matchers.any()),
    ...
);
```

and the magic happens :)

## 5 Simple Logging Facade for Java

---

### 5.1 Simple Logging Facade for Java (SLF4J)

Users that want to use the *SLF4J* package and let *Guice* injects automatically `org.slf4j.Logger` instances, have to add the following dependency in the `pom.xml`:

```
<dependency>
  <groupId>com.google.code.sli4j</groupId>
  <artifactId>sli4j-slf4j</artifactId>
  <version>XX.XX</version>
  <scope>compile</scope>
</dependency>
```

then, when creating the `com.google.inject.Injector`, add the `com.google.code.sli4j.slf4j.Slf4jLoggingModule` module; please take note that users have to specify the classes `com.google.inject.matcher.Matcher` for whom the logging injection has to be applied:

```
import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.code.sli4j.slf4j.Slf4jLoggingModule;
import com.google.inject.matcher.Matchers;
...
Injector injector = Guice.createInjector(new Slf4jLoggingModule(Matchers.any()),
    ...
);
```

and the magic happens :)

#### 5.1.1 Direct SLF4J bindings

The module above uses the `org.slf4j.LoggerFactory` to create `org.slf4j.Logger` instances, but *sli4j* comes with native *SLF4J* bindings, resumed in the following table:

binding	groupId	artifactId	module class
nop	com.google.code.sli4j	sli4j-slf4j-nop	com.google.code.sli4j.slf4j.nop.Slf4jNopLoggingModule
simple	com.google.code.sli4j	sli4j-slf4j-simple	com.google.code.sli4j.slf4j.simple.Slf4jSimpleLoggingModule
log4j12	com.google.code.sli4j	sli4j-slf4j-log4j	com.google.code.sli4j.slf4j.log4j.Slf4jLog4jLoggingModule
jdk14	com.google.code.sli4j	sli4j-slf4j-jdk14	com.google.code.sli4j.slf4j.jdk14.Slf4jJdk14LoggingModule
jcl	com.google.code.sli4j	sli4j-slf4j-jcl	com.google.code.sli4j.slf4j.jcl.Slf4jJclLoggingModule
logback	com.google.code.sli4j	sli4j-slf4j-logback	com.google.code.sli4j.slf4j.logback.Slf4jLogbackLoggingModule

## 6 Extend sli4j

---

### 6.1 Extend sli4j

Exigent users that have the need to integrate not already supported logging framework, can easily do it by following the listed steps:

- 1 add the core dependency in the pom.xml:

```
<dependency>
  <groupId>com.google.code.sli4j</groupId>
  <artifactId>sli4j-core</artifactId>
  <version>XX.XX</version>
  <scope>compile</scope>
</dependency>
```

- 2 Extend the `com.google.code.sli4j.core.AbstractLoggerInjector`, that's the class responsible of creating and injecting the desired `Logger`, specifying the `Logger` type:

```
import java.lang.reflect.Field;
import com.acme.MyLogger;
import com.acme.MyLoggerFactory;
import com.google.code.sli4j.core.AbstractLoggerInjector;
public final class AcmeLoggerInjector extends AbstractLoggerInjector<MyLogger> {
    public AcmeLoggerInjector(Field field) {
        super(field);
    }
    @Override
    protected MyLogger createLogger(Class<?> klass) {
        return MyLoggerFactory.getLog(klass);
    }
}
```

- 3 Extend the `com.google.code.sli4j.core.AbstractLoggingModule`, specifying the `Logger` type and the `com.google.code.sli4j.core.AbstractLoggerInjector` type:

```
import com.acme.MyLogger;
import com.google.code.sli4j.core.AbstractLoggingModule;
import com.google.inject.TypeLiteral;
import com.google.inject.matcher.Matcher;
public final class AcmeLoggingModule extends AbstractLoggingModule<MyLogger> {
    public ACLLoggingModule(Matcher<? super TypeLiteral<?>> matcher) {
        super(matcher, AcmeLoggerInjector.class);
    }
}
```

- 4 Plug your new logging module and enjoy ;)