# SimpleLogging Package

Version 1.6
June 2021

**Gray Watson**

# Table of Contents

# SimpleLogging

Version 1.6 – June 2021

I've written this code into a number of different libraries of mine including ORMLite and I have decided that it is worth it's own release. The goal is a small-ish library that backends to a number of standard logging packages. This allows you to write your code and include log messages without having a fixed dependency on any one logging package. I include this code into my libraries and so they can stay agnostic. I understand that this is the goal of commons-logging as well but I really want the slf4j-style `{}` arguments which aren't supported. This logging code allows you to write messages with the `{}` support and then backend to a number of other logging libraries or you can easily implement your own. It also handles arrays appropriately and supports up to 4 arguments before forcing the caller to pass in an object array.

To get started quickly using SimpleLogging, see Chapter 1 [Quick Start], page 2. There is also a HTML version of this documentation.

Gray Watson `http://256stuff.com/gray/`

# 1 Start Using Quickly

To use SimpleLogging in your code is similar to other logging libraries that you are used to.

```
// usually a logger will be per-class, getLogger() also can take a String label
private static final Logger logger = LoggerFactory.getLogger(MyClass.class);
...

// log trace message with arguments. toString() on the args only called if trace messa
logger.trace("some trace information: {} and {}", arg1, arg2);
...

// NOTE: exception argument comes _before_ the message format to not confuse the argum
logger.error(exception, "http client threw getting URL: {}", url);
```

For somewhat more extensive instructions, see Chapter 2 [Using], page 3.

# 2 Using SimpleLogging

## 2.1 Downloading Jar

To get started with SimpleLogging, you will need to download the jar file. The SimpleLogging release page is the default repository but the jars are also available from the central maven repository.

The code works with Java 6 or later.

## 2.2 Using SimpleLogging in Your Library

To use the SimpleLogging code in your library, just copy the classes into a logging package in your code and rename the package as necessary.

## 2.3 How SimpleLogging Discovers the Logging Backlend

Built into the `LoggerFactory` class is a `LogBackendType` enumerated type which SimpleLogging uses to try to locate what logging packages are on the classpath and therefore should be used.

The following logging implementations will be discovered on the classpath in this order.

1. SLF4J (often paired with logback)
2. Android native Log
3. Apache Commons Logging
4. LOG4J2 (version 2+)
5. LOG4J (older)
6. Local log implementation that can write to a simple file.
7. Java util logging which is usually available in the JRE but never chosen directly.

If you (or the users of your library) want to force the logging type, they can use the following static method.

```
// used to set a specific LoggerFactory.LogBackendType or custom
LoggerFactory.setLogBackendFactory(...);
```

## 2.4 More Usage Examples

The logger

```
// usually a logger will be per-class, getLogger() also can take a String label
private static final Logger logger = LoggerFactory.getLogger(MyClass.class);
...

// log trace message with arguments. toString() on the args only called if trace messa
logger.trace("some trace information: {} and {}", arg1, arg2);
```

```
...

// NOTE: exception argument comes _before_ the message format to not confuse the argum
logger.error(exception, "http client threw getting URL: {}", url);
```

## 2.5 Using With Maven

To use SimpleLogging with maven, include the following dependency in your ‘pom.xml’ file:

```
<dependency>
   <groupId>com.j256.simplelogging</groupId>
   <artifactId>simplelogging</artifactId>
   <version>1.6</version>
</dependency>
```

# 3 Open Source License

This document is part of the SimpleLogging project.

# Index of Concepts