

log4j - Configuration



Complete Python Prime Pack

9 Courses 2 eBooks

Tutorialspoint

[More Detail](#)



Artificial Intelligence & Machine Learning Prime Pack

6 Courses 1 eBooks

Tutorialspoint

[More Detail](#)



Java Prime Pack

9 Courses 2 eBooks

Tutorialspoint

[More Detail](#)

The previous chapter explained the core components of log4j. This chapter explains how you can configure the core components using a configuration file. Configuring log4j involves

assigning the Level, defining Appender, and specifying Layout objects in a configuration file.

The **log4j.properties** file is a log4j configuration file which keeps properties in key-value pairs. By default, the LogManager looks for a file named **log4j.properties** in the **CLASSPATH**.

- ▢ The level of the root logger is defined as **DEBUG**. The **DEBUG** attaches the appender named X to it.
- ▢ Set the appender named X to be a valid appender.
- ▢ Set the layout for the appender X.

log4j.properties Syntax:

Following is the syntax of *log4j.properties* file for an appender X:

```
# Define the root logger with appender X
log4j.rootLogger = DEBUG, X

# Set the appender named X to be a File appender
log4j.appender.X=org.apache.log4j.FileAppender

# Define the layout for X appender
log4j.appender.X.layout=org.apache.log4j.PatternLayout
log4j.appender.X.layout.conversionPattern=%m%n
```

log4j.properties Example

Using the above syntax, we define the following in **log4j.properties** file:

- ▢ The level of the root logger is defined as **DEBUG**, The **DEBUG** appender named **FILE** to it.
- ▢ The appender **FILE** is defined as **org.apache.log4j.FileAppender**. It writes to a file named **log.out** located in the **log** directory.
- ▢ The layout pattern defined is **%m%n**, which means the printed logging message will be followed by a newline character.

```
# Define the root logger with appender file
log4j.rootLogger = DEBUG, FILE

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.conversionPattern=%m%n
```

It is important to note that log4j supports UNIX-style variable substitution such as `${variableName}`.

Debug Level

We have used DEBUG with both the appenders. All the possible options are:

- ☐ TRACE
- ☐ DEBUG
- ☐ INFO
- ☐ WARN
- ☐ ERROR
- ☐ FATAL
- ☐ ALL

These levels are explained later in this tutorial.

Appenders

Apache log4j provides Appender objects which are primarily responsible for printing logging messages to different destinations such as consoles, files, sockets, NT event logs, etc.

Each Appender object has different properties associated with it, and these properties indicate the behavior of that object.

Property	Description
layout	Appender uses the Layout objects and the conversion pattern associated with them to format the logging information.
target	The target may be a console, a file, or another item depending on the appender.
level	The level is required to control the filtration of the log messages.
threshold	Appender can have a threshold level associated with it independent of the logger level. The Appender ignores any logging messages that have a level lower than the threshold level.
filter	The Filter objects can analyze logging information beyond level matching and decide whether logging requests should be handled by a particular Appender or ignored.

We can add an Appender object to a Logger by including the following setting in the configuration file with the following method:

```
log4j.logger.[logger-name]=level, appender1, appender..n
```



You can write same configuration in XML format as follows:

```
<logger name="com.apress.logging.log4j" additivity="false">
  <appender-ref ref="appender1"/>
  <appender-ref ref="appender2"/>
</logger>
```

If you are willing to add Appender object inside your program then you can use following method:

```
public void addAppender(Appender appender);
```

The `addAppender()` method adds an Appender to the Logger object. As the example configuration demonstrates, it is possible to add many Appender objects to a logger in a comma-separated list, each printing logging information to separate destinations.

We have used only one appender *FileAppender* in our example above. All the possible appender options are:

- ☐ AppenderSkeleton
- ☐ AsyncAppender
- ☐ ConsoleAppender
- ☐ DailyRollingFileAppender
- ☐ ExternallyRolledFileAppender
- ☐ FileAppender
- ☐ JDBCAppender
- ☐ JMSAppender
- ☐ LF5Appender
- ☐ NTEventLogAppender
- ☐ NullAppender
- ☐ RollingFileAppender
- ☐ SMTPAppender
- ☐ SocketAppender
- ☐ SocketHubAppender
- ☐ SyslogAppender
- ☐ TelnetAppender
- ☐ WriterAppender

We would cover FileAppender in Logging in Files [↗](#) and JDBC Appender would be covered in Logging in Database [↗](#).

Layout


We have used PatternLayout with our appender. All the possible options are:

- ☐ DateLayout

-  HTMLLayout
-  PatternLayout
-  SimpleLayout
-  XMLLayout

Using HTMLLayout and XMLLayout, you can generate log in HTML and in XML format as well.

Layout Formatting

You would learn how to format a log message in chapter: [Log Formatting](#)  .

