

# LOG4J - INTERVIEW QUESTIONS

## Advertisements

Dear readers, these **log4j Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **log4j**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer –

What is log4j?

log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.

log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages.

log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations, such as a database, file, console, UNIX Syslog, etc.

What are the components of log4j?

log4j has three main components –

- loggers: Responsible for capturing logging information.
- appenders: Responsible for publishing logging information to various preferred destinations.
- layouts: Responsible for formatting logging information in different styles.

What are the features of log4j?

Following are features of log4j –

- It is thread-safe.
- It is optimized for speed.
- It is based on a named logger hierarchy.
- It supports multiple output appenders per logger.
- It supports internationalization.
- It is not restricted to a predefined set of facilities.
- Logging behavior can be set at runtime using a configuration file.
- It is designed to handle Java Exceptions from the start.
- It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL.
- The format of the log output can be easily changed by extending the Layout class.
- The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface.
- It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination.

What are Pros and Cons of Logging?

Following are the Pros and Cons of Logging –

Logging is an important component of the software development. A well-written logging code offers quick debugging, easy maintenance, and structured storage of an application's runtime information.

Logging does have its drawbacks also. It can slow down an application. If too verbose, it can cause scrolling blindness. To alleviate these concerns, log4j is designed to be reliable, fast and extensible.

Since logging is rarely the main focus of an application, the log4j API strives to be simple to understand and to use.

What is the purpose of Logger object?

Logger Object – The top-level layer of log4j architecture is the Logger which provides the Logger object. The Logger object is responsible for capturing logging information and they are stored in a namespace hierarchy.

What is the purpose of Layout object?

Layout Object – The layout layer of log4j architecture provides objects which are used to format logging information in different styles. It provides support to appender objects before publishing logging information.

Layout objects play an important role in publishing logging information in a way that is human-readable and reusable.

What is the purpose of Appender object?

Appender Object – This is a lower-level layer of log4j architecture which provides Appender objects. The Appender object is responsible for publishing logging information to various preferred destinations such as a database, file, console, UNIX Syslog, etc.

What is the purpose of Level object?

Level Object - The Level object defines the granularity and priority of any logging information. There are seven levels of logging defined within the API: OFF, DEBUG, INFO, ERROR, WARN, FATAL, and ALL.

What is the purpose of Filter object?

Filter Object – The Filter object is used to analyze logging information and make further decisions on whether that information should be logged or not. An Appender objects can have several Filter objects associated with them. If logging information is passed to a particular Appender object, all the Filter objects associated with that Appender need to approve the logging information before it can be published to the attached destination.

What is the purpose of ObjectRenderer object?

ObjectRenderer – The ObjectRenderer object is specialized in providing a String representation of different objects passed to the logging framework. This object is used by Layout objects to prepare the final logging information.

What is the purpose of LogManager object?

LogManager – The LogManager object manages the logging framework. It is responsible for reading the initial configuration parameters from a system-wide configuration file or a configuration class.

What is the use of log4j.properties?

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.

What is the purpose of layout object in Appender?

layout – Appender uses the Layout objects and the conversion pattern associated with them to format the logging information.

What is the purpose of target in Appender?

target – The target may be a console, a file, or another item depending on the appender.

What is the purpose of level in Appender?

level – The level is required to control the filtration of the log messages.

What is the purpose of threshold in Appender?

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.

What is the purpose of filter in Appender?

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.

How will you define a root logger with appender file using log4j.properties?

Following syntax defines the root logger with appender file:

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

How will you define a file appender using log4j.properties?

Following syntax defines a file appender –

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

How will you define the layout of file appender using log4j.properties?

Following syntax defines the layout of file appender –

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

How will you create a logger in any class?

Any other named Logger object instance is obtained through the second method by passing the name of the logger. The name of the logger can be any string you can pass, usually a class or a package name as we have used in the last chapter and it is mentioned below –

```
static Logger log = Logger.getLogger(log4jExample.class.getName());
```

How will you print a log message in debug mode?

public void debug(Object message) of Logger class prints messages with the level Level.DEBUG.

How will you print a log message in error mode?

public void error(Object message) of Logger class prints messages with the level Level.ERROR.

How will you print a log message in fatal mode?

public void fatal(Object message) of Logger class prints messages with the level Level.FATAL.

How will you print a log message in info mode?

public void info(Object message) of Logger class prints messages with the level Level.INFO.

How will you print a log message in warn mode?

public void warn(Object message) of Logger class prints messages with the level Level.WARN.

How will you print a log message in trace mode?

public void trace(Object message) of Logger class prints messages with the level Level.TRACE.

What is purpose of ALL log level?

ALL – All levels including custom levels.

What is purpose of DEBUG log level?

DEBUG – Designates fine-grained informational events that are most useful to debug an application.

What is purpose of ERROR log level?

ERROR – Designates error events that might still allow the application to continue running.

What is purpose of FATAL log level?

FATAL – Designates very severe error events that will presumably lead the application to abort.

What is purpose of INFO log level?

INFO – Designates informational messages that highlight the progress of the application at coarse-grained level.

What is purpose of OFF log level?

OFF – The highest possible rank and is intended to turn off logging.

What is purpose of TRACE log level?

TRACE – Designates finer-grained informational events than the DEBUG.

What is purpose of WARN log level?

WARN – Designates potentially harmful situations.

How do Levels Works?

A log request of level p in a logger with level q is enabled if  $p \geq q$ . This rule is at the heart of log4j. It assumes that levels are ordered. For the standard levels, we have ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF.

How will you define a root logger turning DEBUG mode off?

Following syntax defines the root logger with WARN mode turning DEBUG mode off.

```
# Define the root logger with appender file
log = /usr/home/log4j
log4j.rootLogger = WARN, FILE
```

What is the purpose of PatternLayout object?

If you want to generate your logging information in a particular format based on a pattern, then you can use org.apache.log4j.PatternLayout to format your logging information.

The PatternLayout class extends the abstract org.apache.log4j.Layout class and overrides the format() method to structure the logging information according to a supplied pattern.

What is the purpose of c character used in the conversionPattern of PatternLayout object?

c – Used to output the category of the logging event. For example, for the category name "a.b.c" the pattern %c{2} will output "b.c".

What is the purpose of C character used in the conversionPattern of PatternLayout object?

C – Used to output the fully qualified class name of the caller issuing the logging request. For example, for the class name. "org.apache.xyz.SomeClass", the pattern %C{1} will output "SomeClass".

What is the purpose of d character used in the conversionPattern of PatternLayout object?

d – Used to output the date of the logging event. For example, `%d{HH:mm:ss,SSS}` or `%d{dd MMM yyyy HH:mm:ss,SSS}`.

What is the purpose of F character used in the conversionPattern of PatternLayout object?

F – Used to output the file name where the logging request was issued.

What is the purpose of l character used in the conversionPattern of PatternLayout object?

l – Used to output location information of the caller which generated the logging event.

What is the purpose of L character used in the conversionPattern of PatternLayout object?

L – Used to output the line number from where the logging request was issued.

What is the purpose of m character used in the conversionPattern of PatternLayout object?

m – Used to output the application supplied message associated with the logging event.

What is the purpose of M character used in the conversionPattern of PatternLayout object?

M – Used to output the method name where the logging request was issued.

What is the purpose of n character used in the conversionPattern of PatternLayout object?

n – Outputs the platform dependent line separator character or characters.

What is the purpose of p character used in the conversionPattern of PatternLayout object?

p – Used to output the priority of the logging event.

What is the purpose of r character used in the conversionPattern of PatternLayout object?

r – Used to output the number of milliseconds elapsed from the construction of the layout until the creation of the logging event.

What is the purpose of t character used in the conversionPattern of PatternLayout object?

t – Used to output the name of the thread that generated the logging event.

What is the purpose of x character used in the conversionPattern of PatternLayout object?

x – Used to output the NDC (nested diagnostic context) associated with the thread that generated the logging event.

What is the purpose of X character used in the conversionPattern of PatternLayout object?

X – The X conversion character is followed by the key for the MDC. For example, `X{clientIP}` will print the information stored in the MDC against the key `clientIP`.

What is the purpose of % character used in the conversionPattern of PatternLayout object?

% – The literal percent sign. `%%` will print a % sign.

What are format modifiers?

By default, the relevant information is displayed as output as is. However, with the aid of format modifiers, it is possible to change the minimum field width, the maximum field width, and justification.

What is the intent of %20c format modifier?

%20c – Left pad with spaces if the category name is less than 20 characters long.

What is the intent of %-20c format modifier?

%-20c – Right pad with spaces if the category name is less than 20 characters long.

What is the intent of `%.30c` format modifier?

`%.30c` – Truncate from the beginning if the category name is longer than 30 characters.

What is the intent of `%20.30c` format modifier?

`%20.30c` – Left pad with spaces if the category name is shorter than 20 characters. However, if the category name is longer than 30 characters, then truncate from the beginning.

What is the intent of `%-20.30c` format modifier?

`%-20.30c` – Right pad with spaces if the category name is shorter than 20 characters. However, if category name is longer than 30 characters, then truncate from the beginning.

If you want to generate your logging information in an HTML-formatted file, how will you proceed?

If you want to generate your logging information in an HTML-formatted file, then you can use `org.apache.log4j.HTMLLayout` to format your logging information.

The `HTMLLayout` class extends the abstract `org.apache.log4j.Layout` class and overrides the `format()` method from its base class to provide HTML-style formatting.

What kind of information `HTMLLayout` class provides?

It provides the following information to be displayed –

- The time elapsed from the start of the application before a particular logging event was generated.
- The name of the thread that invoked the logging request.
- The level associated with this logging request.
- The name of the logger and logging message.
- The optional location information for the program file and the line number from which this logging was invoked.

How will you set the content type of html generated using `HTMLLayout`?

`HTMLLayout.setContentType(String)` – Sets the content type of the HTML content. Default is `text/html`.

How will you set the location information for the logging event using `HTMLLayout`?

`HTMLLayout.setLocationInfo(String)` – Sets the location information for the logging event. Default is `false`.

How will you set the title of html page generated using `HTMLLayout`?

`HTMLLayout.setTitle(String)` – Sets the title for the HTML file. Default is `log4j Log Messages`.

What is the purpose of `immediateFlush` configuration of `FileAppender`?

`immediateFlush` – This flag is by default set to `true`, which means the output stream to the file being flushed with each append operation.

What is the purpose of `encoding` configuration of `FileAppender`?

`encoding` – It is possible to use any character-encoding. By default, it is the platform-specific encoding scheme.

What is the purpose of `threshold` configuration of `FileAppender`?

`threshold` – The threshold level for this appender.

What is the purpose of `Filename` configuration of `FileAppender`?

`Filename` – The name of the log file.

What is the purpose of `fileAppend` configuration of `FileAppender`?

fileAppend – This is by default set to true, which means the logging information being appended to the end of the same file.

What is the purpose of bufferedIO configuration of FileAppender?

bufferedIO – This flag indicates whether we need buffered writing enabled. By default, it is set to false.

What is the purpose of bufferSize configuration of FileAppender?

bufferSize – If buffered I/O is enabled, it indicates the buffer size. By default, it is set to 8kb.

How will you configure immediate flush to true using log4j.properties?

Following code configures immediate flush to true –

```
# Set the immediate flush to true (default)
log4j.appender.FILE.ImmediateFlush=true
```

How will you set the threshold to debug mode using log4j.properties?

Following code sets the threshold to debug mode –

```
# Set the threshold to debug mode
log4j.appender.FILE.Threshold=debug
```

How will you set the append to false, overwrite using log4j.properties?

Following code sets the append to false, overwrite –

```
# Set the append to false, overwrite
log4j.appender.FILE.Append=false
```

If you want to write your logging information into multiple files then how will you proceed?

To write your logging information into multiple files, you would have to use org.apache.log4j.RollingFileAppender class which extends the FileAppender class and inherits all its properties.

What is the purpose of maxFileSize property of RollingFileAppender class?

This is the critical size of the file above which the file will be rolled.

What is default value of maxFileSize property of RollingFileAppender class?

Default value is 10 MB.

What is the purpose of maxBackupIndex property of RollingFileAppender class?

This property denotes the number of backup files to be created.

What is default value of maxBackupIndex property of RollingFileAppender class?

Default value is 1.

How will you configure a RollingFileAppender using log4j.properties?

Following code configures a RollingFileAppender –

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

How will you configure maximum file size before rollover using log4j.properties?

Following code configures maximum file size before rollover –

```
# Set the maximum file size before rollover
log4j.appender.FILE.MaxFileSize=5KB
```

How will you configure maximum files to be used to log data using log4j.properties?

Following code configures maximum files to be used –

```
# Set the the backup index
log4j.appender.FILE.MaxBackupIndex=2
```

What happens if logs exceeding the maximum size while using RollingFileAppender?

A new log file will be created.

What happens if last log file reaches the maximum size while using RollingFileAppender?

Once the last log file reaches the maximum size, the first log file will be erased and thereafter, all the logging information will be rolled back to the first log file.

How will you generate your log files on a daily basis?

To write your logging information into files on a daily basis, you would have to use org.apache.log4j.DailyRollingFileAppender class which extends the FileAppender class and inherits all its properties.

What is the purpose of DatePattern property of DailyRollingFileAppender class?

This indicates when to roll over the file and the naming convention to be followed. By default, roll over is performed at midnight each day.

How will you configure your log to roll over at the end of each month and at the beginning of the next month?

'.' yyyy-MM – Roll over at the end of each month and at the beginning of the next month.

How will you configure your log to roll over at midnight each day?

'.' yyyy-MM-dd – Roll over at midnight each day. This is the default value.

How will you configure your log to roll over at midday and midnight of each day?

'.' yyyy-MM-dd-a – Roll over at midday and midnight of each day.

How will you configure your log to roll over at the top of every hour?

'.' yyyy-MM-dd-HH – Roll over at the top of every hour.

How will you configure your log to roll over every minute?

'.' yyyy-MM-dd-HH-mm – Roll over every minute.

How will you configure your log to roll over on the first day of each week depending upon the locale?

'.' yyyy-ww – Roll over on the first day of each week depending upon the locale.

How will you configure a DailyRollingFileAppender using log4j.properties?

Following code configures a DailyRollingFileAppender –

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

How will you set the DatePattern using log4j.properties?

Following code configures a DatePattern –



```
# Set the DatePattern
log4j.appender.FILE.DatePattern = '.' yyyy-MM-dd-a
```

How will you put the logs in database using log4j?

The log4j API provides the org.apache.log4j.jdbc.JDBCAppender object, which can put logging information in a specified database.

What is the purpose of driver configuration of JDBCAppender?

driver – Sets the driver class to the specified string. If no driver class is specified, it defaults to sun.jdbc.odbc.JdbcOdbcDriver.

What is the purpose of password configuration of JDBCAppender?

password – Sets the database password.

What is the purpose of sql configuration of JDBCAppender?

sql – Specifies the SQL statement to be executed every time a logging event occurs. This could be INSERT, UPDATE, or DELETE.

What is the purpose of URL configuration of JDBCAppender?

URL – Sets the JDBC URL.

What is the purpose of user configuration of JDBCAppender?

user – Sets the database user name.

## What is Next?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)