**Log4j useful information and configurations:**

* Log4j is reliable, fast and flexible logging framework (APIs) written in Java.
* Can be ported to C, C++, C#, Python, Ruby languages. Log4j is a third part API used for the purpose of **creating logging information at runtime**.

With **log4j** it is possible to enable **logging at runtime** **without modifying the application binary**. The log4j package is designed so that ***these statements can remain in shipped code without incurring a heavy performance cost***.

Log4j has **three main components**:

* **logger:** is an inbuild class in Log4j API, responsible for capturing logging information.
* **Appender**: is an interface, responsible for publishing logging information to various preferred destinations.
* [creating text files, xml, html format in folder and also create pattern layouts for designing logging information]
* **Layout:** is an interface, responsible for formatting logging information in different styles.

**Important Notes:**

* logging information means it will create log file with information such as time of execution date of execution, logging failures and success of execution etc, execution can be for development codes or testing scripts.
* Getting log information helps us to understand the root cause of failure or success of a test script or a development code.

**How it works:**

**Logger** -- > are **objects** with given **name through** which **application makes logging calls**. Every logger has one or more **appender tied to it**.

**Appender** - - > is the **destination** and **determines** where **logs are recorded**. eg: **Console, file, db** etc. [JDBCAppender – to write in database, FileAppender – to a file, ConsoleAppender – to write messages to Console, SMTPAppender -to write to mail]

**Layout** - - > is **tied to an appender** and determines the **fields** that are **logged in messages**. eg: **plain text, xml, html formats** etc. [ PatternLayout, SimpleLayout, XMLLayout, HTMLLayout ]

**Pattern** - - > is **tied to a layout** and determines the **fields** that are **logged in messages besides the given string**. eg: **timestamps, class name, method name, line number** etc.

**Log Level** - - > This is important concept we need to know about log4j log-level. When **we write log in application**. **We specify the criticality or level of log**. eg. **Ordinary log, important log or critical log**. when **we configure log4j to print logs** which are ordinary or a higher level, so we can choose at run-time, the verbosity of logs.

Log4j has **7 different logging levels** and have hierarchy levels

1. All – highest logging level
2. Trace
3. Debug
4. Info
5. Warn
6. Error
7. Fatal - lowest logging level. [ Note: if we configure the **logging level to be Debug**, then we **can create logs for Debug, Info, Warn, Error, Fatal**. We **cannot** use **Trace or All**].

\* To **create logs,** we need the **log4j.properties** file or **log4j.xml** file. **[Note: these two files contains configurations to give our logging information placed in project folder].**

**Log4j maven dependency :**

<!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

# We have defined root logger

# above we will mention the level in root logger

log4j.rootlogger=INFO.CONSOLE,R,HTML.TTCC

# We define the appender

# the code appender interface is used for publishing captured

log4j.appendeer.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.R=org.apache.log4j.RollingFileAppender

log4j.appender.TTCC=org.apache.log4j.RollingFileAppender

log4j.appender.HTML=org.apache.log4j.FileAppender

# We define log file location

# the code for the destination folders and files are stored.

log4j.appender.R.File=./log/testlog.log

log4j.appender.TTCC.File=./log/testlog1.log

log4j.appender.HTML.File=./log/application.html

# We define the layout and pattern

log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout

log4j.appender.CONSOLE.layout.ConversionPattern= %sp [%t] (%F:%L) - %m%n

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d - %c -%p -%m%n

log4j.appender.TTCC.layout=org.apache.log4j.TTCCLayout

log4j.appender.TTCC.layout.DateFormat=ISO8601

log4j.appender.HTML.layout=org.apache.log4j.HTMLLayout

log4j.appender.HTML.layout.Title=Application log

log4j.appender.HTML.layout.Locationinfo=true

The below code must be added to capture logs.

// code to capture the logging information which should be written in the java class and pass the class name.

**Logger log=Logger.getLogger(“<javaclassname.class>”);**

// to tell log4j to find the external log4j.properties, we define by configuring as very first line in class.

**BasicConfigurator.confugure();**

// we have to configure to PropertyConfiiguration, it helps to recognize properties file

**PropertyConfigurator.configure(“log4j.properties”);**

// this is the way we have to write code in java class file to get log info wherever we want in code.

**log.info(“ some meaningful message”);**

**1.Console Logging**: configure log4j to **write logs** on **console** using log4j.properties file.

* **log4.rootLogger=INFO, loggerId**

**initializing root logger** with **log level INFO** with **name loggerId**

* **log4.appender.loggerId=org.apache.log4.ConsoleAppender**

setting an **appender** for **rootlogger** using its name loggerId. **This appender is ConsoleAppender**, **which logs messages to console**.

* **log4j.appender.loggerId.layout=org.log4j.PatternLayout**

setting **layout** for the **log statements**.

**2**.**File Logging**: configure log4j to **write logs** in a **file** using log4j.properties file.

* **log4j.rootlogger=INFO, loggerId**

**initializing root logger** with **log level INFO** with **name loggerId**

* **log4j.appender.loggerid=org.apache.log4j.FileAppender**

setting an **appender** for **rootlogger** using its name loggerId. **This appender is FileAppender**, **which logs messages to file**.

* **log4j.loggerId.layout=org.apache.log4j.PatternLayout**

setting **layout** for the **log statements**.

* **log4j.loggerId.layout.ConversionPattern=%d [%t] %-5p (%F : %L) - %m %n**

setting conversion pattern layout along with timestamp, thread name log level etc.

* **log4j.appender.loggerIf.File=c:\users\sballa\log1.log**

setting the appender to write to a file.

**Conversion pattern:**

**%d** - - >stands for **current timestamp**.

**%t** - - >stands for **java thread name.**

**%p** - - > stands for **log level.**

**%F** - - > stands for **name of class that logged the message**.

**%L** - - > stands for **line number where the message is logged.**

**%m** - - >stands for **logged message string.**

**%n** - - > stands for **new line character.**

**Custom Pattern:**

**%C** - - >complete **class name** that has **invoked the log**.

**%d** - - > **timestamp invoked** can be further customized **java’s DateFormat style** like **%d(yyyy-MM-dd – HH:mm:ss, SSS)** which incidentally is the default used**.**

**%l - ->** the **source file name, method name** and **line number** where **log request is made.**

**%M** -- > **the method name** that **makes the log request.**

**%r** - - > the **time elapsed between start of the application** and the **log request in milliseconds.**

**3.Making log files rotate based on size using log4j.properties:**

* **log4j.rootlogger=INFO, loggerId**

**initializing root logger** with **log level INFO** with **name loggerId**

* **log4j.appender.loggerid=org.apache.log4j.RollingFileAppender**

setting an **appender** for **rootlogger** using its name loggerId. **This appender is RollingFileAppender**, **which logs messages to file.** If the file size exceeds given sizelog4j.properties file will rename the current log+1 index and replace by new file to log.

* **log4j.loggerId.layout=org.apache.log4j.PatternLayout**

setting **layout** for the **log statements**.

* **log4j.loggerId.layout.ConversionPattern=%d [%t] %-5p (%F : %L) - %m %n**

setting conversion pattern layout along with timestamp, thread name log level etc.

* **log4j.appender.loggerId.File=c:\users\sballa\log1.log**

setting the appender to write to a file.

* **log4j.appender.loggerId.MaxFileSize=100KB**

setting the log file size. ( can be used MB or KB file sizes.

* **log4j.appender.loggerId.MaxBackupIndex=5**

setting the number of backup file can be created.

**4.Making log files rotate everyday using log4j.properties: (from above code]**

* **log4j.appender.loggerid=org.apache.log4j.DailyRollingFileAppender**

and add DatePattern – eg. file rolling per month.

* log4j.appender.loggerId.DatePattern=’-’yyyyMMdd’.log’

and add DatePattern – eg. file rolling per hour.

* log4j.appender.loggerId.DatePattern=’-’yyyyMMddHH’.log’

and add DatePattern – eg. file rolling per minute.

* log4j.appender.loggerId.DatePattern=’-’yyyyMMddHHmm’.log’

**5.configure multiple appender using log4j.properties:**

log4j.rootLogger=INFO, loggerid1, loggerid2

log4j.appender.loggerid1=org.apache.log4j.ConsoleAppender

log4j.appender.loggerid1=org.apache.log4j.PatternLayout

log4j.appender.loggerid2=org.apache.log4j.RollingFileAppender

log4j.appender.loggerid2=org.apache.log4j.PatternLayout

log4j.appender.loggerid2.layout.ConversionPattern=%d [%t] %5p (%F : %L) - %m %n

log4j.appender.loggerid2.File=c:\users\sballa\log.log

**6.configure multiple logger using log4j.properties**

log4j.rootLogger=INFO, loggerid

log4j.appender.loggerid=org.apache.log4j.ConsoleAppender

log4j.appender.loggerid.layout=org.apache.log4j.PatternLayout

log4j.logger.com.chakra.log4j.BasicUse=DEBUG, loggerid1

log4j.appender.loggerid1=org.apache.log4j.RollingAppender

log4j.appender.loggerid1.layout=org.apache.log4j.PatternLayout

log4j.appender.loggerid1.ConversionPattern=%d [ %d] % -5p ( %F : % L) -- %m%n

log4j.appender.loggerid1.File=c:\users\sballa\mylogs.log

**7.configure log appending across application restarts instead of overwrite:**

log4j.rootLogger=INFO, loggerId

log4j.appender.loggerId=org.apache.log4j.RollingFileAppender

log4j.appender.loggerId.layout=org.apache.log4j.PatternLayout

log4j.appender.loggerId.layout.ConversionPattern=%d [%t] %-5p (%F:%L) - %m%n

log4j.appender.loggerId.File=exampleee.log

log4j.appender.loggerId.append=false