**Log4j API:**

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**.

It has **three main components**:

**logger:** is an inbuild class in Log4j API, responsible for capturing logging information.

**appenders**: is an interface, responsible for publishing logging information to various preferred destinations.

[ creating text files, html format in folder and also create pattern layouts for designing logging information]

**layouts:** is an interface, responsible for formatting logging information in different styles.

**Note:**

* 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.

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

**All – highest logging level**

**Traces**

**Debug**

**Info**

**Warn**

**Error**

**Fatal - lowest logging level**

[if we configure the logging level to be Debug, then we can create logs for Debug, Info, Warn, Error, Fatal. We can not use Traces are All].

* \* For **creating logs** we need the **log4j.properties** file or **log4j.xml** file.

**[Note: these two files contain 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>

* Create a properties file ( log4j.properties)

# 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”);

// 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”);