==========

Topic: **Log4J**

==========

1. Document prepared by: **Rajat Verma**
    a. https://www.linkedin.com/in/rajat-v-3b0685128/
    b. https://github.com/rajatt95
    c. https://rajatt95.github.io/

========== Testing - Log4J ==========

Log4J:

1. Links:
    a. https://logging.apache.org/log4j/2.x/
    b. https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core
2. Log4j is a reliable, fast, and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.
3. Log4j is a popular logging package written in Java. log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages.
4. It is an open-source logging API for java.
5. Used for Logging purposes.
6. It helps to generate Log files in various output targets.
7. Main components:
    a. Loggers: Responsible for capturing logging information.
    b. Appenders: Responsible for publishing logging information to various preferred destinations.
    c. Layouts: Responsible for formatting and logging information in different styles.

—----------

Log4J features:

1. Thread-safe.
2. Optimized for speed.
3. Supports multiple output appenders per logger.
4. Supports Internationalization.
5. It is not restricted to a predefined set of facilities.
6. Logging behavior can be set at runtime using a configuration file.
7. It is designed to handle Java Exceptions from the start.

8. It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL.
9. The format of the log output can be easily changed by extending the *Layout* class.
10. The target of the log output, as well as the writing strategy, can be altered by implementations of the Appender interface.
11. It is a fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination.

—----------

Logging:

1. Logging is a powerful aid for understanding and debugging the runtime behavior of the programs. Simply the logging means some way to indicate the state of the system at runtime. Logs are used to capture and persist the important data and make it available for analysis at any point in time.
2. logging means some way to indicate the state of the system at runtime. Logs are used to capture and persist the important data and make it available for analysis at any point in time.

—----------

Advantages of Logging:

1. Quick Debugging
2. Problem Diagnosis
3. Easy Maintenance
4. Cost and Time Savings

—----------

Log File:

1. The process of storing execution flow step by step is known as **Logging.**
2. Any file extension that is **.log** will be considered as a Logfile**.**
3. Format: HTML, Text, etc.

—----------

Logging levels:

1. All
2. Debug
3. Info
4. Warn
5. Error
6. Fatal
7. Off
8. Trace

—----------

1.

Level	Description
ALL	All levels including custom levels.
DEBUG	Designates fine-grained informational events that are most useful to debug an application.
INFO	Designates informational messages that highlight the progress of the application at coarse-grained level.
WARN	Designates potentially harmful situations.
ERROR	Designates error events that might still allow the application to continue running.
FATAL	Designates very severe error events that will presumably lead the application to abort.
OFF	The highest possible rank and is intended to turn off logging.
TRACE	Designates finer-grained informational events than the DEBUG.

—----------

2.

			x: Visible				
	FATAL	ERROR	WARN	INFO	DEBUG	TRACE	ALL
OFF							
FATAL	x						
ERROR	x	x					
WARN	x	x	x				
INFO	x	x	x	x			
DEBUG	x	x	x	x	x		
TRACE	x	x	x	x	x	x	
ALL	x	x	x	x	x	x	x

—----------

3.

Detail Included in Logs

Logging Level	Debug statements	Informational messages	Warning statements	Error statements	Fatal statements
ALL	Included	Included	Included	Included	Included
DEBUG	Included	Included	Included	Included	Included
INFO	Excluded	Included	Included	Included	Included
WARN	Excluded	Excluded	Included	Included	Included
ERROR	Excluded	Excluded	Excluded	Included	Included
FATAL	Excluded	Excluded	Excluded	Excluded	Included
OFF	Excluded	Excluded	Excluded	Excluded	Excluded

CODE - START

---

1. **Log4j.properties:**

a.

```
#Here, we define Root Logger
log4j.rootLogger=INFO,CONSOLE,R,HTML,TTCC

#Here, we define the Appender
log4j.appender.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

#Here, we define Log file location
log4j.appender.R.File=./log/testlog.log
log4j.appender.TTCC.File=./log/testlog1.log
log4j.appender.HTML.File=./log/application.html

#Here, we define the Layout and Pattern
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%5p [%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
```

---

2. Test script:

```
package _02_Log4J;

import java.util.concurrent.TimeUnit;

public class _01_Log4J {

    private static final String LOG4J_PROPERTIES = ".//src//test//resources//log4j_config//log4j.properties";

    public static void main(String[] args) {

        // We need to create a Logger instance, so we need to pass Class name
        // for which, we want to create Log file
        Logger logger = Logger.getLogger(_01_Log4J.class);
        // Logger logger = Logger.getLogger("_01_Log4J");

        // Configure log4j.properties file
        PropertyConfigurator.configure(LOG4J_PROPERTIES);

        // Open Chrome Browser Instance
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();
        logger.info("Chrome Browser opened");

        // Maximize the Window
        driver.manage().window().maximize();
        logger.info("Browser Window maximized");

        // Set Implicit Wait
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        logger.info("Impicit Wait given");

        // Open URL
        driver.get("https://en-gb.facebook.com/");
        logger.info("Application launched");

        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        // Quit Driver
        driver.quit();
        logger.info("Application closed");

    }
}
```

---

3. Reports/Logs :

Log session start time Sat Apr 23 10:41:49 IST 2022

Time	Thread	Level	Category	File:Line	Message
0	main	INFO	_02_Log4J._01_Log4J	_01_Log4J.java:29	Chrome Browser opened
141	main	INFO	_02_Log4J._01_Log4J	_01_Log4J.java:33	Browser Window maximiz
149	main	INFO	_02_Log4J._01_Log4J	_01_Log4J.java:37	Impicit Wait given
3085	main	INFO	_02_Log4J._01_Log4J	_01_Log4J.java:41	Application launched
6182	main	INFO	_02_Log4J._01_Log4J	_01_Log4J.java:50	Application closed

a.

testlog.log

```
2022-04-23 10:42:32,447 - _02_Log4J._01_Log4J -INFO - Chrome Browser opened
2022-04-23 10:42:32,588 - _02_Log4J._01_Log4J -INFO - Browser Window maximized
2022-04-23 10:42:32,596 - _02_Log4J._01_Log4J -INFO - Impicit Wait given
2022-04-23 10:42:35,532 - _02_Log4J._01_Log4J -INFO - Application launched
2022-04-23 10:42:38,629 - _02_Log4J._01_Log4J -INFO - Application closed
```

b.

CODE - END

==========