Logging using Log4j

- A log is a message that we are recording or generating for each transaction we do. We can analyze the logs to check what went correct or wrong. Suppose if any system closes abruptly then by analyzing the logs, we can get into the root cause of the failure.
- Thus logs are generated in every development cycle. In the same way, we can also generate logs in our Selenium code for testing before and after each test condition or test statement, to check if all are working as expected.
- To generate these log files in Selenium code, we use ***log4j*** framework provided by Apache. With this framework, we can generate our customized logs.
- We can generate logs in 2 ways:
  - Using log4j.properties file
  - Using log4j.xml file
- These files will contain the configuration about what kind of logs you want to generate. You can use any one of them. If you wish to use both then log4j.xml will be given higher precedence. The preferred way to generate logs is using the properties file, so here we will explore more about generating through properties file only.

Implementation Of log4j

- Download the log4j jar file and add it in your project build path. Create ***log4j.properties*** file and add the properties file parallel to your source folder when you are using standalone java application.
- Log4j.properties file is a configuration file that stores values in the key-value pair.
- It contains 3 main components:
  - Loggers: Captures logging information.
  - Appenders: Publish logging information to a different preferred destination like consoles, files, sockets, NT event logs, etc.
  - Layouts: Format logging information in different styles like HTML, XML Layout, etc.

Syntax Of log4j.properties File

- Define the root logger with the logging level INFO and appender X [appender can be any consoles, files, sockets, NT event logs].

  log4j.rootLogger = INFO, X
- Set the appender named X to be a File Appender.

  log4j.appender.X = org.apache.log4j.FileAppender

- Define the layout for the X appender.

  log4j.appender.X.layout = org.apache.log4j.PatternLayout
  log4j.appender.X.layout.conversionPattern = %m%n

log4j.properties Example : Create a log4j.properties file, referring to the above syntax

- initialize root logger with level INFO and print it in the console using stdout and fout.

  log4j.rootLogger=INFO,stdout,fout

- add a ConsoleAppender to the logger stdout to write to the console.

  log4j.appender.stdout=org.apache.log4j.ConsoleAppender

- use a simple message format layout pattern defined is %m%n, which prints logging messages in a newline.

  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
  log4j.appender.stdout.layout.ConversionPattern=%m%n

- add a FileAppender to the logger fout.

  log4j.appender.fout=org.apache.log4j.FileAppender

- The appender FILE is defined as org.apache.log4j.FileAppender. It writes to a file named SoftwareTestingHelp.

  log4j.appender.fout.File=google.log

- use a more detailed message pattern.

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

  log4j.appender.fout.layout.ConversionPattern=%p\t%d{ISO8601}\t%r\t%c\t[%t]\t%m%n

  - **Different Levels Of Logging**
    - DEBUG
    - INFO
    - WARN
    - ERROR
    - FATAL
  - Each level has its own priority. Suppose if we use “DEBUG” level, then it will log all level messages like INFO >> WARN >> ERROR >> FATAL.

- Suppose if we use “ERROR” level, then it will ignore DEBUG >> INFO >> WARN and it will only log ERROR >> FATAL.
- In all these levels we need to define in our properties file. The logs will be generated, depending upon our configuration.

Steps to Integrate Log4j into existing automation Framework

- Create a ***log4j.properties*** file under ***src/main/resources*** package and write the statements as below.

```
# Set logging level
log4j.rootCategory=debug, console, file

# Appender which writes to console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{MM-dd-yyyy HH:mm:ss} %F %-5p [%t] %c{2} %L - %m%n

# Appender which writes to a file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=./logs/google.log

# Defining maximum size of a log file
log4j.appender.file.MaxFileSize=5mb
log4j.appender.file.MaxBackupIndex=5
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ISO8601} %5p [%t] %c{1}:%L - %m%n
```

- Create instance of the logger as below.
  public static Logger logger = Logger.getLogger(BaseTest.class);