Log4j

- Log4j is fast, flexible and Reliable logging framework.
- under apache software license.
- why?
  - * open-source
  - * Save logs in file, console, database
  - * can be used for small and large projects.
  - * Replacement of System.out.println
  - * formatting the output/Error/log of the application.
  - * Enable or disable some category of logs.
    like error, warn, info, debug, fatal ....
- log4j.Properties is a configuration file contains key-value pairs
- Components of log4j

  (1) Loggers : logging informations

  (2) Appenders : Appending logs on file or db or console

  (3) Layout : Responsible for formatting logging information in different forms.

- Levels of loggers : defines priority of log

  (1) All : logges all the process of execution.

  (2) DEBUG : used to print information about debugging

  (3) INFO : information about progress of application process

  (4) WARN : Print about faulty and unexpected behaviours.

(5) ERROR: Print Error messages with all to continue System Execution.

(6) FATAL: Print System critical information, which are causing application to crash.

(7) OFF : No logging.

* Appenders:

The appender basically grabs information from the logger and writes to the file or any other storage location.

Appenders in log4j

(1) FileAppender :→ This will append the log to a file

(2) RollingFileAppender :→ it will perform the same function as FileAppender, But users will be able to specify the maximum size of file. once the limit is exceded, the appender will create another file to write message.

(3) DailyRollingFileAppender :→ specify the frequency by which the message are to be written to the file. (daily basis storage).

(4) ConsoleAppenders :→ Simply writes log in console.

---

defining root Logger

log = /user/home/log4j

log4j.rootLogger = DEBUG, FILE
category

* Creating logger for class

step (1) : Logger log = Logger.getLogger (classname.class);

step (2) : propertyConfigurator.configure ("log4j.properties");

* Level wise ?

All < debug < Info < warn < Error < fatal < off

Ex: root logger turning debug mode off

log4j.rootCategory = warn, file, console

---

* Layout

The layout is where the format in which log message will appear is decided.

(1) Pattern layout :

Based on the pattern provided the logs will displayed.

it takes default conversion pattern, if not specified.

(2) HTML layout : in the form of HTML Table.

(3) xml layout : in the form of xml format.

Pattern layout

%c in pattern layout

%c{2} Fully qualified name of logger issuing class

%c{1} Same class name

%d output date & logging

%L output line number

%m output application supplied message.

%M method name if any

%n output platform dependent line separator char

%p Priority of logging object

%t thread name generated logging event.

Note: For sample log4j.properties file refer the file provided in classroom

```
#set level

log4j.rootCategory=debug,console,file


 #appender which write on 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} %p [%t] %c{2} %L -
%m%n


 #appender writes to file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=D:\\Sep_Mrng_2021_JavaAutomation\\MavenTest\\target\\logs\\applogs.log


#defining max size and format in file

log4j.appender.file.MaxFileSize=10MB

log4j.appender.file.MaxBackupIndex=10

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

log4j.appender.file.layout.ConversionPattern=%d{ISO8601} %p [%t] %c{1}:%L -%m%n

log4j.appender.file.Append=true
```