Log4j : is framework ( framework is semi-build application )

**Importance of Log4j :**

* Can write data on console
* Can write data to a file
* Can work with multiple files

Addressed increasing file size program whenever we want to know how the test is running.

**Three main Components of Log4j :**

**Loggers**: responsible for capturing logging info.

**Appenders**: responsible for publishing logging info to various preferred destinations

**Layouts**: responsible for formatting logging info in different styles.

It contains Control-flags to control OFF ALL

**Logging levels**

ALL – All levels including custom levels

TRACE – Designates finer-grained informational events than debug

DEBUG -- Designates finer-grained informational events that are most useful to debug

INFO -- Designates finer-grained informational messages that highlights the progress of the application

WARN-- Designates potentially harmful situations

ERROR -- Designates error events that might still allow application to continue running. informational events than debug

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

**Logger** is pre-defined java class

Logger logger= … instantiate this

logger.trace(“”) means trace mode

logger.debug“”) means debug mode – Dev level

logger.info(“”) means info mode -- Production level

logger.warn(“”)

logger.error(“”)

logger.fatal(“”)

**Appenders**

JDBCAppender – to write in database

FileAppender – to a file

ConsoleAppender – to write messages to Console

SMTPAppender

**Layouts**

PatternLayout

SimpleLayout

XMLLayout

HTMLLayout

To Implement Log4j (Configuration)

* Download latest log4j distribution / use maven configuration – by adding dependency in POM.xml
* Add log4j’s jar library into your program’s classpath
* Create log4j’s configuration
* Initialize log4j with the configuration
* Create a logger
* Put logging statements into your code.

Dependencies for Maven Project

<**dependency**>  
 <**groupId**>log4j</**groupId**>  
 <**artifactId**>log4j</**artifactId**>  
 <**version**>1.2.17</**version**>  
 </**dependency**>

1. Default name: log4.xml – and should be in the Classpath

How to write a data to file using Log4j :

Logger Object:

private static final Logger logger=Logger.getLogger(Classname.class);

BasicConfigurator.*configure* ();

>>**Properties file configuration**

>>To configure log4j from external .properties file, invoke the static method configure() of the class PropertyConfigurator

PropertyConfigurator.configure(String configFilename)

**Properties files – with Console Appender >>>> name should be Log4j.properties**

log4j.rootLogger=Debug, **console**

**# Console configiration**

log4j.appender.**console**=org.apache.log4j.ConsoleAppender

log4j.appender.**console**.layout=org.apache.log4j.PatternLayout

log4j.appender.**console**.layout.ConversionPattern=%d[dd-mm-yyyy hh:mm:ss z] %-5p %c %x -%m%n

**Properties files – with File Appender**

log4j.rootLogger=Debug, **file**

**# File Configuration**

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

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

log4j.appender.**file**.layout=appLogs.log

log4j.appender.**file**.layout.ConversionPattern=%d[dd-mm-yyyy hh:mm:ss z] %-5p %c %x -%m%n

And change the code > from Basic configuration to Properties file.

PropertyConfigurator.configure(System.getProperty(“user.dir”+”log4j.propertiestring”); configFilename)