**Log 4j**

Log4j built with three main components which work together to enable log messages according to message type and level, and to control at runtime how these messages are formatted and where they are reported.

***Loggers***: which are responsible for taking the input of logging information and display the logging information as per your needs.

***log4j.rootLogger***

***Appenders*** : Which are responsible to display the logging information. The name of of the appender is "console" and this is the name that is used to refer to the appender in the rest of the configuration file. The class to use for the appender is org.apache.log4j.ConsoleAppender. It can be used to set standard output console / text file / html etc***.***

***log4j.appender.***

***Layout*** : The console appender also has a layout element defined which uses org.apache.log4j.PatternLayout . Looking at the javadoc for  
[PatternLayout](https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html) , the layout.ConversionPattern method takes a string describing the layout for messages and Layout is used for formatting the log file output.

***log4j.appender.console.layout***

***log4j.appender.file.layout***

***ALL***- Which has the lowest possible rank and is intended to turn on all logging.

***DEBUG*** is the lowest restricted logging level and should write everything we need to debug an application, this java logging mode should only be used on development and testing environment and we should avoid debug mode in production environment which impacts performance of the application.

***INFO*** is more restricted than DEBUG logging level and we should log messages to know information such has when the server has been started, when the connections are established etc in INFO level logging in java.

***WARN*** : We should use WARNING for minor user-level exceptions that are not expected during normal processing, anything that can potentially cause application being strange etc. We can monitor health of the application and work on these warning messages. We should define in such a way Viewing a log should give quick insight into early hints at the root cause of the error. Warnings should be used in a simple manner so that they don't become meaningless or difficult to understand.

***ERROR*** is like there is a problem, definitely that should be investigated. It should be defined in such a way to quickly identify the failure that might have resulted in a cascade of additional errors. Ex: An intermittent error, such as lost session even after waiting for some x seconds. etc

***FATAL*** These Fatal errors should occur if something is missing or a situation occurs for which the application can simply not continue or application can't do any more useful work. Possible examples are a missing required config.file or when an exception pops up' and is caught by an unhandled exception handler.

// To start logging add the line inside the class (e.g. MainClass)

static Logger logger = Logger.getLogger(MainClass.class);

// Set up a simple configuration that logs on the console.

BasicConfigurator.configure();

// Set up a configuration from property file (e.g. log4j.properties).

PropertyConfigurator.configure("resource/log4j/log4j.properties");

