**Log4J**

Log4j is open source tool, given by Apache, only for Java projects.

Log4j is used to know the status of a Java application at its execution time.

**Components of Log4j**

We have 3 components in Log4j

* **Logger**
* **Appender**
* **Layout**

**Logger** component enables Log4j in java class, so we need to create a Logger object one per java class.

* To get an object of logger class, we need to call the static factory method.
* Logger methods are used to generate a log statements in java class.
* We must create a Logger object right after class name.

**Syntax:** private static final Logger log = Logger.getLogger(YourClassName.class.getName());

* **Logger** class contains 5 methods and these methods are used to print the status of our application.
* **debug()**
* **info()**
* **warn()**
* **error()**
* **fatal()**

**Method n**ames are different but all are same. For human identification purpose names are different, else these 5 methods will print one text message only.

As a programmer we should know where we need to use which method.

**Priority Order:**

debug < info < warn < error < fatal

**Appender** Job is to write the messages into external file or database or smtp.

* Logger class generates log statements under different levels, this Appender takes these log statements and stores in some files or database
* Appender is an interface

In log4j we have different Appender implementation classes

* FileAppender [ writing into a file ]
* ConsoleAppender [ Writing into console ]
* JDBCAppender [ For Databases ]
* SMTPAppender [ Mails ]
* SocketAppender [ For remote storage ]

Again in FileAppender we have 2 more

* RollingFileAppender
* DailyRollingFileAppender

**Layout** specifies the format of log statements which are written into destination repository by appender.

We have different type of layout classes in log4j

* SimpleLayout
* PatternLayout
* HTMLLayout
* XMLLayout

**Slf4J(Simple Logging façade for Java)**

**Slf4J** is not an implementation of logging framework, it is an abstraction of all logging frameworks in java similar to Log4j. Therefore, you cannot compare both (Log4j vs Slf4j).

**Priority Order:**

Trace< debug < info < warn < error < fatal

**Syntax:** private static final Logger logger = LoggerFactory.getLogger(classname.class);

Logging abstraction is always preferable than logging framework. If you use a logging abstraction (SLF4J in particular) you can migrate to any logging framework at the time of deployment without opting for single dependency.

