Introduction to LOG4J

Introduction to Log4j

Log4j 1.2 Introduction

- One of Java’s promising logging package is Log4j
- Initially in 1996, it was designed as a tracing and logging API for the project called E.U. SEMPER (Secure Electronic Marketplace for Europe)
- Later with evolution under openSource Apache software license Log4j was distributed

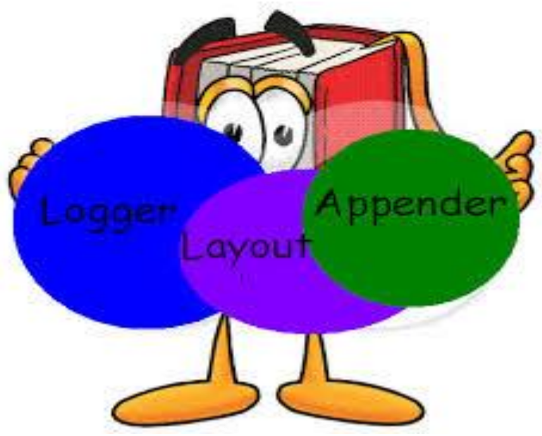
Logo of Log4j

Features of Log4J

- Reliable
- Fast
- Extensible
- Thread Safe
- Internationalization support
- Behavior can be set at runtime using a config file
- Can handle Java Exceptions

Log4j components

- There are three main components in Log4j which aid the development team to log messages, categorizing and formatting it
- These components are:
  - Loggers: This captures the logging information
  - Appenders: This publishes the logging information to various destinations
  - Layouts: This formats the logging information with styles



Logger, Appenders and Layouts

LOGGER

- What we want to log?
  [Errors, Warnings, Informations etc.]

APPENDER

- Where we want to log?
  [Console, log file etc.]

LAYOUT

- How do we display in log?
  [Simple, Pattern etc.]

Loggers

- Loggers is basically an object which capture the messages and its metadata for logging

(Metadata here means: date, time, and other details of the event occurred)

- These messages are then passed to the logging framework
- The top logger of hierarchy is called as Root Logger
- Root Logger is said to behave in certain exceptional ways like
  - Always it exists
  - But by name retrieval is not possible
  - And never its level can be set to null

Instantiating the root logger:

Invoking the static Logger.getRootLogger() method instantiates the Root Logger

Logger Naming

- Why do we name the Logger?

Loggers are named by the developers for easy retrieval and referencing in the application

- A logger name can be anything decided by the development team
- But the best strategy is a fully qualified name of the class

Logger Naming

Benefits of Naming Loggers:

- **Simple to implement**
- **Simple to explain to new developers**
- **It automatically mirrors your application's own modular desig n**
- **If required It can be further refined**
- **Printing the logger automatically gives information on the locality of the log statement, as the output bears the name of the generating logger**

Logger Naming

Invoking the static **Logger.getLogger(String name)** method instantiates the other Loggers

- Here desired logger’s name is passed in as parameter
- Eg: Logger.getLogger(“test”);

Summary

- Introduction to Log4J
- Features of Log4J
- Log4J components
- Loggers, Appenders and Layouts

Thank You