April 2012

MyLogger Manual

If you are new to MyLogger, here is a short description on what is it and why should we care about it:

MyLogger is a flexible logging library, an open source project from Aloysius. Using MyLogger, we can replace the debugging print line statements, like *System.out.println("Value is " + someVariable),* with a configurable logging statement like logger.debug("Value is " + someVariable), which can be switched off in production version.

How to add MyLogger logger support to your project?

1. Download the *MyLogger.jar* in your project class path. For Java web application, you can place the jar file in *WEB-INF/lib* folder. For Java applications, you can place the jar in any folder, but remember to add the folder to your classpath.

2. Next we need to configure the MyLogger library to our requirements. MyLogger reads its configurations from MyLogger.properties file placed in the CLASSPATH.

3. Every MyLogger.properties file defines the following three things, mainly:
   - MyLogger Appender –It could be a simple Console appender which writes the log messages to stdout (screen) or a file appender, which sends the log messages to a log file.
   - MyLogger Formatter – is nothing but how the log message is formatted. This format is very simlar to C languages's *printf* function formatting.
   - Logfile – Logger tells that the log messages from these packages should go to some appender which will log the message.
   - MyLogger Level -Loggers *may* be assigned levels. The set of possible levels, that is: `TRACE, DEBUG, INFO, WARN, ERROR and FATAL` are defined in the myjava.mylog.LoggerLevel class.

     Logging requests are made by invoking one of the printing methods of a logger instance. These printing methods are trace, debug, info, warn, error, fatal. By definition, the printing method determines the level of a logging request. For example, if c is a logger instance, then the statement c.info("..") is a logging request of level INFO.

     A logging request is said to be *enabled* if its level is higher than or equal to the level of its logger. Otherwise, the request is said to be *disabled*. A logger without an assigned level will inherit one from the hierarchy.The standard levels, we have TRACE < DEBUG < INFO< WARN < ERROR < FATAL.

4. Below is a sample MyLogger.properties for configuring the console appender for your project.

```
logger.level=5
logger.filename=C:\\MyJava\\LoggerClient\\logs\\MyLogger1.log
logger.appender=FILE,CONSOLE
logger.format=%-5s:%n
```

In the above properties file, we define the log level to trace, the logfile to MyLogger1.log, then the appender to file and console and the format to `%-5s:%n`

5. In any of your Java file, add the below lines, in order to start logging:

```
LoggerConfig.configureLog(“Yourpath\\MyLogger.properties”);

private static Logger logger = Logger.getLogger(MyclassName.class);

logger.debug("this is a sample log message.");
```

6. Now you are done. Run your application and you should be seeing the log messages coming in your console window.