**Logging Messages - Methods**

To log a message using a java.util.logging.Logger, you call one of its many logging methods. Among these are:

log (Level level, String message);

log (Level level, String message, Object param1);

log (Level level, String message, Object[] params);

log (Level level, String message, Throwable t);

log (LogRecord record);

logp (Level level, String sourceClass, String sourceMethod, String msg);

logp (Level level, String sourceClass, String sourceMethod, String msg, Object param1);

logp (Level level, String sourceClass, String sourceMethod, String msg, Object[] params);

logp (Level level, String sourceClass, String sourceMethod, String msg, Throwable t);

logrb(Level level, String sourceClass, String sourceMethod, String bundle, String msg);

logrb(Level level, String sourceClass, String sourceMethod, String bundle, String msg, Object param1);

logrb(Level level, String sourceClass, String sourceMethod, String bundle, String msg, Object[] params);

logrb(Level level, String sourceClass, String sourceMethod, String bundle, String msg, Throwable t);

entering(String sourceClass, String sourceMethod);

entering(String sourceClass, String sourceMethod, Object param1);

entering(String sourceClass, String sourceMethod, Object[] params);

exiting (String sourceClass, String sourceMethod);

exiting (String sourceClass, String sourceMethod, Object result);

fine (String message);

finer (String message);

finest (String message);

config (String message);

info (String message);

warning (String message);

severe (String message);

throwing(String sourceClass, String sourceMethod, Throwable t);

I am not going to explain every single of these methods in detail. They are explained in the JavaDoc's for Java. But, I am going to explain their purpose. Knowing their purpose you can most likely figure out the rest, with help from the JavaDoc.

**The log() Methods**

The log() group of methods will log a message at a certain log level. The log level is passed as parameter. Use one of the java.util.logging.Level constants as parameter. Log level is covered in more detail in its own text.

Some of the log() methods can take object parameters. These object parameters are inserted into the log message, before it is being logged. The merging of object parameters into the message is only performed, if the message is not filtered out, either by a Filter, or because of too low log level. This improves performance in the cases where the message is filtered out.

Here is a log() example:

Logger logger = Logger.getLogger("myLogger");

logger.log(Level.SEVERE, "Hello logging");

And here is what is logged to the console (default log destination) :

08-01-2012 14:10:43 logging.LoggingExamples main

SEVERE: Hello logging

Here is an example that inserts a parameter into the message:

logger.log(Level.SEVERE, "Hello logging: {0} ", "P1");

And here is what is being logged:

08-01-2012 14:45:12 logging.LoggingExamples main

SEVERE: Hello logging: P1

Notice how the object parameter value P1 is inserted at the place in the log message where the {0} is located. The0 is the index of the object parameter to insert.

Here is an example that logs a message with multiple object parameters to be inserted into the log message:

logger.log(Level.SEVERE, "Hello logging: {0}, {1}",

new Object[] {"P1", "P2"});

Here is what is being logged:

08-01-2012 14:45:12 logging.LoggingExamples main

SEVERE: Hello logging: P1, P2

Notice again how the object parameters are inserted into the log message instead of the {0} and {1} tokens. As mentioned earlier, the number inside the token refers to the index of the object parameter to insert, in the object parameter array passed to the log() message.

Here is an example that logs a Throwable:

logger.log(Level.SEVERE, "Hello logging",

new RuntimeException("Error"));

Here is what is being logged (use the scrollbar to see it all):

08-01-2012 14:54:29 logging.LoggingExamples main

SEVERE: Hello logging

java.lang.RuntimeException: Error

at logging.LoggingExamples.main(LoggingExamples.java:18)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at com.intellij.rt.execution.application.AppMain.main(AppMain.java:120)

**The logp() Methods**

The logp() methods work like the log() methods, except each method take an extra two parameters: ThesourceClass and sourceMethod parameter.

These two parameters are intended to tell from what class and method the log message originated. In other words, which class and method was the "source" of the log message.

**The logrb() Methods**

The logrb() methods work like the log() methods too, except they can obtain the log messages from a resource bundle. A resource bundle is a set of texts containing key, value pairs, like a property file. Each file contains the same set of keys, with values in different languages. Resource bundles are an internationalization feature, and I won't cover it in great detail here.

Here is a logrb() example:

logger.logrb(Level.SEVERE, "logging.LoggingExamples", "main",

"resources.myresources", "key1");

This example looks up a message in the resource bundle named resources.myresources by the key key1. If the resource bundle does not contain a key with that name (key1), the value itself is logged as a message. In this example then the value "key1" would have been logged as message, if no key named "key1" had existed in the resource bundle.

Here is what has logged:

08-01-2012 17:14:39 logging.LoggingExamples main

SEVERE: This is message 1

Here is what the resource bundle (property file) looks like:

key1 : This is message 1

key2 : this is message 2

The Java Doc says nothing about how you localize the messages using a Locale. How to select the language of the ResourceBundle, in other words.

**The Last Log Methods**

The Logger also has the following methods for logging:

entering(String sourceClass, String sourceMethod);

entering(String sourceClass, String sourceMethod, Object param1);

entering(String sourceClass, String sourceMethod, Object[] params);

exiting (String sourceClass, String sourceMethod);

exiting (String sourceClass, String sourceMethod, Object result);

fine (String message);

finer (String message);

finest (String message);

config (String message);

info (String message);

warning (String message);

severe (String message);

Each of these methods corresponds to a log level. For instance, finest(), finer(), fine(), info(),warning() and severe() each corresponds to one of the log levels. Logging message using one of these methods corresponds to calling the log() method