LOG4J - LOGGING METHODS

Logger class provides a variety of methods to handle logging activities. The Logger class does not allow us to instantiate a new Logger instance but it provides two static methods for obtaining a Logger object –

- **public static Logger getRootLogger();**
- **public static Logger getLogger(String name);**

The first of the two methods returns the application instance's root logger and it does not have a name.

Any other named Logger object instance is obtained through the second method by passing the name of the logger. The name of the logger can be any string you can pass, usually a class or a package name as we have used in the last chapter and it is mentioned below –

```
static Logger log = Logger.getLogger(log4jExample.class.getName());
```

Logging Methods

Once we obtain an instance of a named logger, we can use several methods of the logger to log messages. The Logger class has the following methods for printing the logging information.

#	Methods and Description
1	**public void debug(Object message)** It prints messages with the level Level.DEBUG.
2	**public void error(Object message)** It prints messages with the level Level.ERROR.
3	**public void fatal(Object message)** It prints messages with the level Level.FATAL.
4	**public void info(Object message)** It prints messages with the level Level.INFO.
5	**public void warn(Object message)** It prints messages with the level Level.WARN.
6	**public void trace(Object message)** It prints messages with the level Level.TRACE.

All the levels are defined in the **org.apache.log4j.Level** class and any of the above mentioned methods can be called as follows –

```
import org.apache.log4j.Logger;

public class LogClass {
   private static org.apache.log4j.Logger log = Logger.getLogger(LogClass.class);

   public static void main(String[] args) {

      log.trace("Trace Message!");
      log.debug("Debug Message!");
      log.info("Info Message!");
      log.warn("Warn Message!");
      log.error("Error Message!");
      log.fatal("Fatal Message!");
   }
}
```

When you compile and run **LogClass** program, it would generate the following result –

```
Debug Message!
Info Message!
Warn Message!
Error Message!
Fatal Message!
```

All the debug messages make more sense when they are used in combination with levels. We will cover levels in the next chapter and then, you would have a good understanding of how to use these methods in combination with different levels of debugging.