

Java™ Logging Overview

OTUG – Java SIG

Donald Griffing



Agenda

- Introduction to Logging
- Java™ Logging Components
- Using Java™ Logging



Introduction to Logging

- The ability to log state information is a critical part of any application.
- Java™ Logging API became built into the language with the release of J2SE 1.4 (JSR 47).
- Lumberjack is an open source implementation of the API for the 1.2 and 1.3 JDKs is available on SourceForge.
- The Apache Logging Services Project (log4j) is widely used.

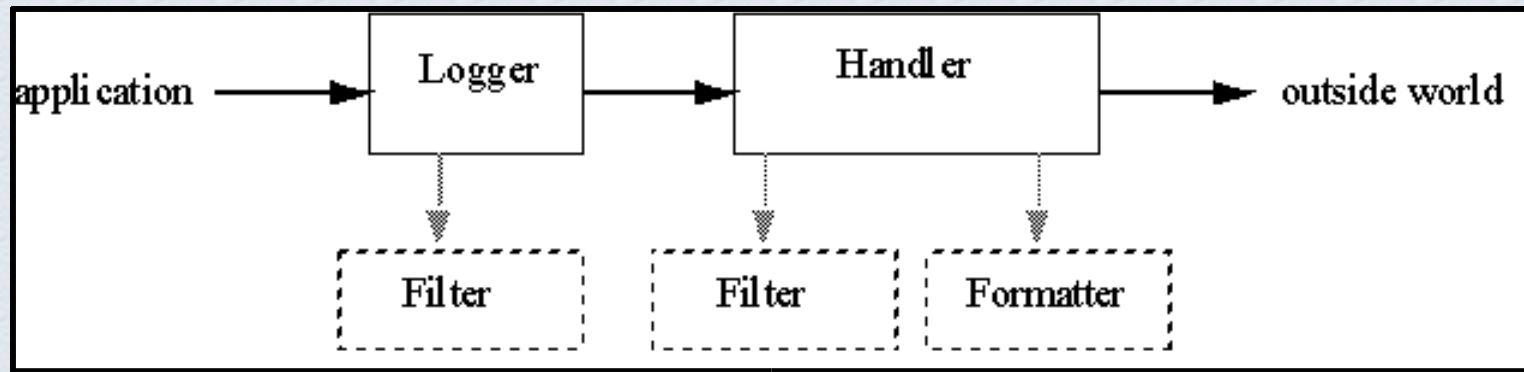


Java™ Logging Components

- Logger objects – main entry points into the logging API.
- Handler objects – controls how messages are published.
- Filters objects – controls if the messages are published.
- Formatter objects – controls what the published log information will look like.



Control Flow



Source: Java™ Logging APIs,
Sun Microsystems, Inc.



Logger Creation

- Create logger using the Logger static method: `getLogger(String loggerName)`.
 - `private static Logger logger = Logger.getLogger(MyClass.class.getName());`
 - `private static Logger logger = Logger.getLogger("com.foobar");`
- The logger namespace is hierarchical and is managed by the `LogManager`.
- The logger name “” (empty string) is the root logger.



Logging Levels

- Level class defines a set of standard logging levels that can be used to control logging output.
- Ranging from highest to lowest, these are: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST.
- It is possible to define additional logging levels by subclassing Level.



Logging Messages

- `log()` methods that take a log level, a message string, and optionally some parameters to the message string. The framework will make a "best effort" to determine which class and method called into the logging method.
- `logp()` methods that take an explicit source class name and method name.
- `logrb()` method that take an explicit resource bundle name.



Logging Messages – Continued

- Convenience methods for tracing method entries, method returns, and throwing exceptions.
- Convenience methods to log a simple string at a given log level . The framework will make a "best effort" to determine which class and method called into the logging method.



Handlers

- `MemoryHandler` – sends the log messages to a memory buffer.
- `StreamHandler` – sends the log messages to an output stream.
- `ConsoleHandler` – sends messages to `System.err`.
- `FileHandler` – sends messages to a file.
- `SocketHandler` – sends messages to a socket.
- It is possible to define additional logging levels by subclassing `Handler` or one of the above.



Formatters

- The API also includes two standard Formatters:
 - SimpleFormatter: Writes brief "human-readable" summaries of log records.
 - XMLFormatter: Writes detailed XML-structured information.
- It is possible to define additional logging levels by subclassing Formatter.



Example & Demo

March 23, 2004

First Degree Solutions, LLC



Issues

- Security – access to local resources.
- Multithreaded applications – methods are not synchronized allowing for messages out of order.
- DTD only available in documentation (JDK 1.4.1_03)



References

- Java™ Logging APIs, Sun Microsystems, Inc.,
<http://java.sun.com/j2se/1.4.2/docs/guide/>
- An Introduction to the Java Logging API,
Brian R. Gilstrap, O'Reilly Media, Inc.,
<http://www.onjava.com/pub/a/onjava/2002>



References – Continued

- Logging in J2SE 1.4, Thomas Paul, JavaRanch's NewsLetter, <http://www.javaranch.com/newsletter/May2>
- The Java Logging API for JDKs prior to 1.4, <http://javalogging.sourceforge.net/>
- Using Log4j with JBoss, Scott Stark, JBoss Group, LLC

