Logging

Devdatta Kulkarni

Logging

- Why do we need logging?
  - Log ``interesting’’ events in an application
    - Application configuration messages
    - Informational debug messages
    - Diagnostic messages
    - Warnings
    - Exceptions

Logging approaches

- Use System.out.println()

- What are the issues?
  - Not fine-grained enough
    - Cannot selectively control which messages are printed
  - Not flexible enough
    - Cannot send/redirect the messages to different “syncs”
      - Only sync is the standard output stream

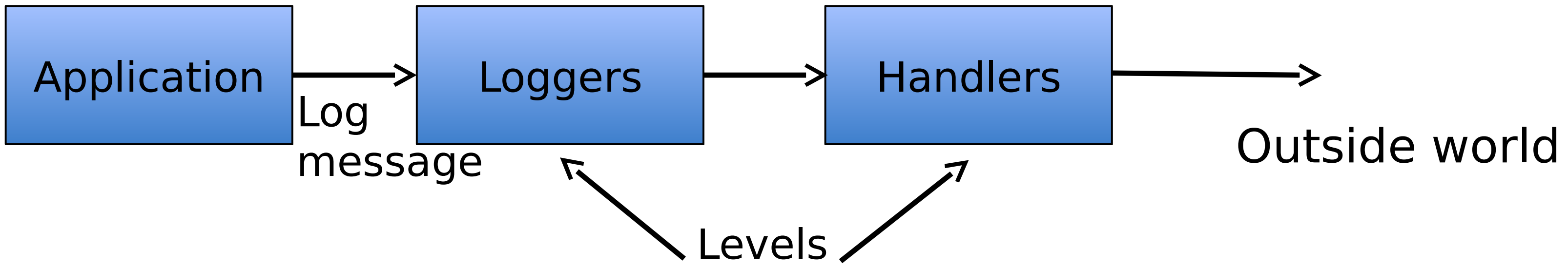
Logging requirements

- Fine-grain control
  - Should be able to support different categories of log messages

- Flexibility
  - When to log?
  - Where to log?

Java Logging Architecture

- Loggers
  - Generate log records
- Handlers
  - Send log records to different destinations



http://docs.oracle.com/javase/7/docs/api/java/util/logging/Logger.html

https://jcp.org/aboutJava/communityprocess/review/jsr047/spec.pdf

What is a logging level?

- A logging level defines relative *order* for log messages
- Logging Levels in Java’s default Logging package (java.util.logging):
  - SEVERE
  - WARNING
  - INFO
  - CONFIG
  - FINE
  - FINER
  - FINEST

Descending order of level

Logging Levels

- Levels are associated with
  - Messages
  - Loggers
    - If set to null, level is inherited from the parent
  - Handlers

- Logging algorithm:
  - A log message is sent to the Logger
  - If message’s level >= logger’s level then the logger passes the message to the handler
  - If message’s level >= handler’s level then the handler sends the message to appenders for output

Logging Levels

- Levels are associated with:
  - Messages
  - Loggers
    - If set to null, level is inherited from the parent
  - Handlers
    - StreamHandler
    - ConsoleHandler
    - FileHandler
    - SocketHandler
    - MemoryHandler

Logging Algorithm details

- Applications make logging calls on Logger
- Logger allocates LogRecord objects which are passed to Handler for publication
- The LogManager class keeps track of a set of global Handlers
- By default all Loggers send their output to this standard set of global Handlers
- Loggers may also be configured to ignore the global Handler list and/or to send output to specific target Handlers.

Ref section 2.1 of
https://jcp.org/aboutJava/communityprocess/review/jsr047/spec.pdf

Distinction between Loggers and Handlers

- Loggers
  - Create LogRecord
  - Sends LogRecords to the default set of global handlers
  - If Logging is enabled and message passes the level check, Logger is careful to minimize execution costs before passing LogRecord into the Handler
- Handlers
  - Localization and formatting of LogRecord (relatively expensive task)
  - Send LogRecords to output

Loggers and Handlers

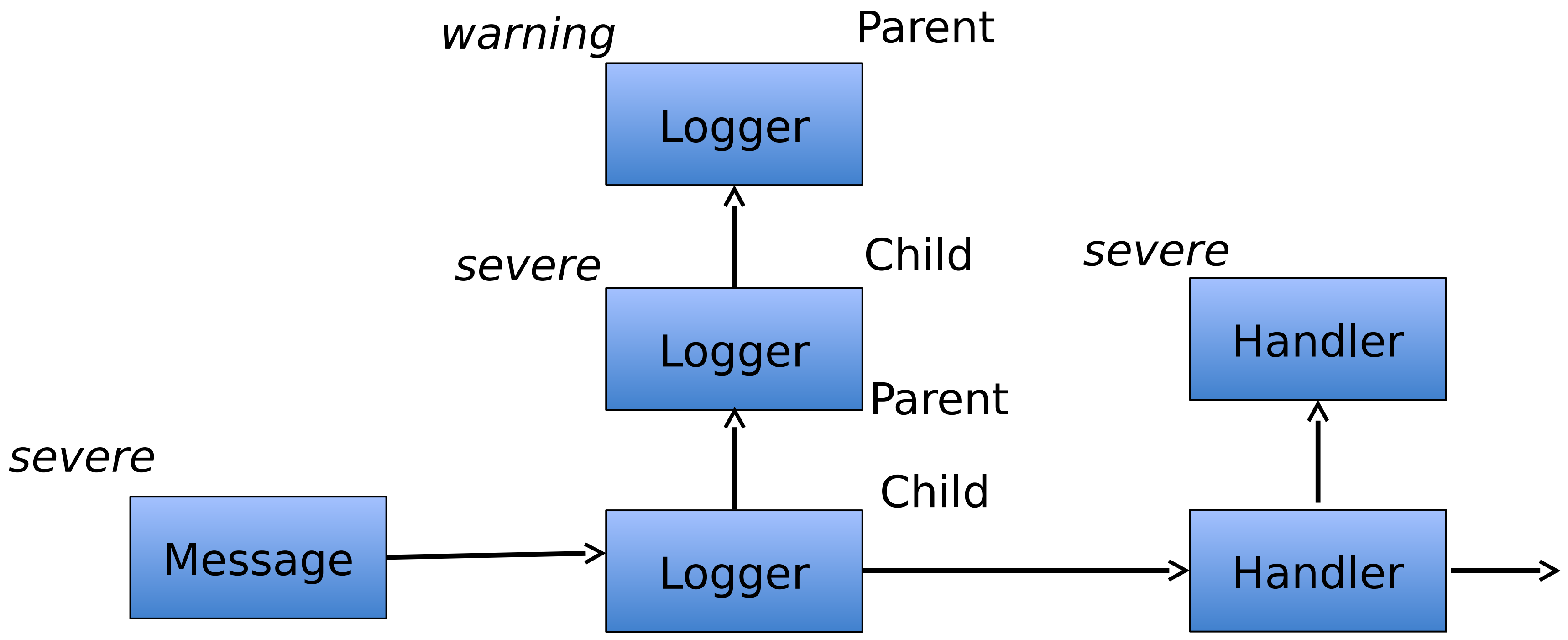
Why perform level checking and filtering in both, loggers and handlers?

- For performance reason
  - If no level checking and filtering is done at Logger, then all messages may need to be sent to all the handlers as there might be more than one handlers each with different destinations
  - If no level checking and filtering in Handler, then all the messages would need to be formatted for output which is an expensive operation

Logging Hierarchy

- Loggers are arranged in a hierarchy
  - The hierarchy is defined based on the names of the loggers
    - The naming hierarchy follows the ``dotted'' notation
      - *cs378.examples.TestLogging* is child of *cs378.examples*
    - If a Class does not have a Logger defined for it, the Logger for its parent is inherited by the Class
      - Root system logger is parent of all loggers
    - A logger will publish the log message to its parent's handlers

Logging example



Logging: Config.properties

Mac
- /Library/Java/JavaVirtualMachines/<JDK Home>/Contents/Home/jre/lib/logging.properties

Ubuntu
- /usr/lib/jvm/java-8-oracle/jre/lib/logging.properties

Specify custom logging.properties
- Use “java.util.logging.config.file” flag

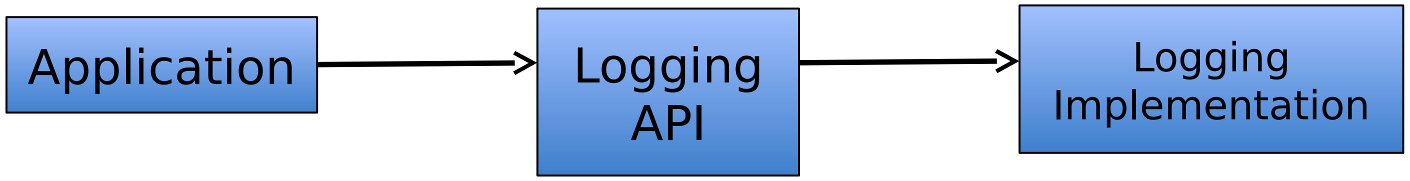
-D java.util.logging.config.file=
/Users/devdatta.kulkarni/Documents/UT/ModernWebApps/logging-
example/logging.properties

Logging Examples

- Example 1: getLoggerAndHandlerLevels()
- Example 2: testPrintInfoMessage()
- Example 3: testPrintConfigMessageFailure()
- Example 4: testPrintConfigMessageLoggerAndHandlerLevelSet()

Logging Framework

- Logging API vs Implementation
  - Swap out implementation without having to change your code



Popular Logging APIs

- Apache commons
  - http://commons.apache.org/proper/commons-logging/
- Simple Logging Façade for Java
  - http://www.slf4j.org/

Logging APIs and Logging Frameworks

- The logging APIs provide ``adapters'' to convert the logging API events to the logging framework
  - Commons logging provides adapters for:
    - Avalon, java.util.logging, Log4j 1.2, and LogKit
  - SLF4J provides adapters for:
    - Commons logging, java.util.logging, and Log4j 1.2

Log4j 2

- http://logging.apache.org/log4j/2.x/
- Provides both API and implementation

Logging Appenders

- Where are logs written?
  - Console
  - Flat Files
  - Sockets
  - SMTP and SMS
  - Databases

References

- https://jcp.org/aboutJava/communityprocess/review/jsr047/spec.pdf
- http://www.onjava.com/pub/a/onjava/2002/06/19/log.html?page=2