latex input: mmd-article-header Title: Java Logging Notes Author: Ethan C. Petuchowski Base Header Level: 1 latex mode: memoir Keywords: logging, java, monitoring, operations CSS: http://fletcherpenney.net/css/document.css xhtml header: copyright: 2016 Ethan Petuchowski latex input: mmd-natbib-plain latex input: mmd-article-begin-doc latex footer: mmd-memoir-footer

Frameworks vs Stdlib

- Java has its own basic logging API, but alternatives are more popular
    - Alternative frameworks include **Log4j** and Logback
    - All of them serve to "deliver logs from your application to a destination"
- **SLF4J** and Apache Commons Logging "provide abstraction layers which decouple your code from the underlying logging framework, allowing you to switch between logging frameworks" [Loggly]
    - This also helps integrate projects that use different frameworks

Architecture

Required

- **Loggers** -- capture events and pass them to the appropriate appender
- **Appenders/Handlers** -- deliver log events to a destination
- **Layouts/Formatters** -- formats data in each log event before an appender sends it to the output

Loggers

- This is what you call methods on in your code
- Maybe you have multiple Loggers for different types of events
    - Or even a nested heirarchy of Loggers
    - This could be useful if you are working on a specific piece of code and only want to turn on debug-level logging for that or a few classes
- First you get a logger by name, and if it doesn't exist yet, it is created
    - General-case best-practice is to name it `getClass.getName` (Scala) or `MyClass.class.getName())` (Java)
- You call methods like `logger.warning("text")`
- You set the threshold ON THIS LOGGER like `logger.setLevel(Level.WARNING)`

Appenders

- You can add appenders to specific loggers
- Appenders might go to a rolling file, stdout, stderr, graphana, email, *syslog* etc.

Layouts

- E.g. plain text, HTML, JSON, serialized
- `PatternLayout` lets you specify a text pattern
    - The default is `"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"`

Optional

- **Filters** -- futher specify whether the Appender should be used for this specific log entry

Configuration

- Most commonly, configuration is set using external configuration files loaded by the framework at runtime
- Log4j can be configured in `log4j2.yaml` files; that's nice

Side Track: Syslog

- Consolidates logs from multiple source into a single location
- Listens on (UDP) port (no authentication)
- Manages a database of log entries
- Provides query functionality
- Provides alerting functionality

References

- Loggly ultimate guide: Java (awesome)