USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Logging

CS 272 Software Development

Department of Computer Science
https://www.cs.usfca.edu/

Motivation

- Using a debugger is complicated in some settings
  - Multithreading, distributed computing
- Using `println` statements are not ideal
  - Easy to create, cumbersome to remove
- Using a custom Debug classes are not ideal
  - Easy to disable, but inefficient and inflexible

Beginner's Guide to Quick Start Debugging (in Eclipse)

Logging

- Next-level `println` debugging
  - Usually output to a log file instead of console
- Configurable without modifying code
  - Can configure **what** (errors, warnings, debug), **where** (console, file), and **how** (threads, line numbers, etc.)
- Usually have separate API and implementation

Logging Options in Java

- java.util.logging · https://docs.oracle.com/en/java/javase/16/core/java-logging-overview.html
- Simple Logging Facade for Java (SLF4J) · http://www.slf4j.org/
- LOGBack · http://logback.qos.ch/
- Apache Log4j2 · https://logging.apache.org/log4j/2.x/

Apache Log4j2

- A third-party library provided by Apache
- Supports multithreaded or distributed environments
- Efficient and flexible logging API and implementation
- Configurable without modifying source code
- Supports lambda expressions for log messages

https://logging.apache.org/log4j/2.x/

Simple Log4j2 Example

```
private static final Logger logger =
        LogManager.getLogger();

public static void main(String[] args) {
    logger.info("Hello, World!");
}
```

https://logging.apache.org/log4j/2.x/manual/api.html

Logging Levels

- **TRACE**
  - Used for fine-grained debug messages
  - Usually not shown or used unless really necessary
- **DEBUG**
  - Used for normal debug messages
  - Most commonly used level for logging

https://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- **INFO**
  - Used for informational messages
  - Often used for major events that were successful
- **WARN**
  - Used when something concerning happened and there *might* be a problem

https://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- **ERROR**
  - Used when an possibly recoverable error occurred
  - Nearly always shown to the user
- **FATAL**
  - Used when an irrecoverable error occurred
  - Often used when about to exit prematurely

https://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- **ALL**
  - Turns on all levels of logging, including TRACE
  - Used to turn on all logging for debugging purposes
- **OFF**
  - Turns off all levels of logging, including FATAL
  - Used to disable logging, speeding up code

https://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

● TRACE	«	lowest level, rarely used
● DEBUG	«	most common
● INFO	«	informational
● WARN	«	warnings
● ERROR	«	errors/exceptions
● FATAL	«	highest level, rarely used

https://logging.apache.org/log4j/2.x/manual/architecture.html

Configurable Output

- Configured via an XML, JSON, or YAML file
- Controls **what** information is output
- Controls **which classes** produce output
- Controls **which levels** are output
- Controls **where** messages are output

https://logging.apache.org/log4j/2.x/manual/configuration.html

Logging Configuration

- Appenders
  - Controls **where** log messages are output
  - Commonly the console and a log file
- Layouts
  - Control **what** information is output for an appender
  - Commonly the timestamp, level, and message

https://logging.apache.org/log4j/2.x/manual/architecture.html

Logging Configuration

- Loggers
  - Identified by a name (often class name)
  - Specify **level** of message to send to an appender
- Root Logger
  - Default logger, always accessible
  - Outputs ERROR messages to console by default

https://logging.apache.org/log4j/2.x/manual/architecture.html

Sample Configuration

```xml
<Configuration status="warn">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%level: %message %n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

Pattern Layout

- **%level** : level of the logging event
    - %level{length=1} to use a single letter
    - %level{lowerCase=true} to convert to lowercase
    - %level{WARN=Warning, ...} to change label
- **%message** : log message
- **%n** : platform dependent line separator (\n or \r\n)

https://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

Pattern Layout

- **%date{pattern}** : timestamp for logging event
  - %d{HH:mm:ss:SSS} to output just time
- **%thread** : thread name (useful later)
- **%location** : location where logging event created
  - Expensive operation, use with caution
  - See also %logger{}, %class{}, %method, and %line

https://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

Pattern Layout

Example Pattern

```
[%date{HH:mm:ss:SSS} %-5level{lowerCase=true}]
%file@%line %t: %m%n
```

Example Output

```
[12:05:53:145 debug] CharacterCounter.java@181 main:
Counting characters in file "src".
```

https://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Software Development
Department of Computer Science

Professor Sophie Engle
sjengle.cs.usfca.edu