JAVA LOGGING STANDARDS

Istvan Fodor

Overview

- Terminology
- Logging Architecture
- Example Use Cases

Terminology I

- Logging Level / Log Level
  - *Levels let us control the detail of our logging. Typically 5 levels:* `trace, debug, info, warn, error.`
  - *Message priority increases from trace to error: for example when we set a global log level to* `info,` *we want to see* `info, warn` *and* `error` *level messages.*
  - *Slf4J example:*
    - `log.info(”Application startup finished at {}”, new Date());`

Terminology II

- Configuration
  - *The configuration defines what log messages get logged (logger) and where (appender) and what gets ignored.*
  - *Most libraries use a file that gets read from the root of the classpath on application startup. Programmatic configuration is also possible, but not recommended, as that is a lot harder to change once the application is deployed.*
  - *Logback: logback.xml*
  - *Log4J2: log4j2.xml, log4j2.json, log4j2.yaml*

Terminology III

- Logger
  - *Created in Java, configured in the configuration file.*
  - *On the Java side it provides the entry point for logging. On the configuration side we get to configure the output format, logging level and other features.*
- Appender
  - *Defined in the configuration, works as a sink, that writes messages to a particular target (console, file, email, database, etc).*
  - *Most of the message customization is in appenders, such as format, filters, additional logged variables, etc.*

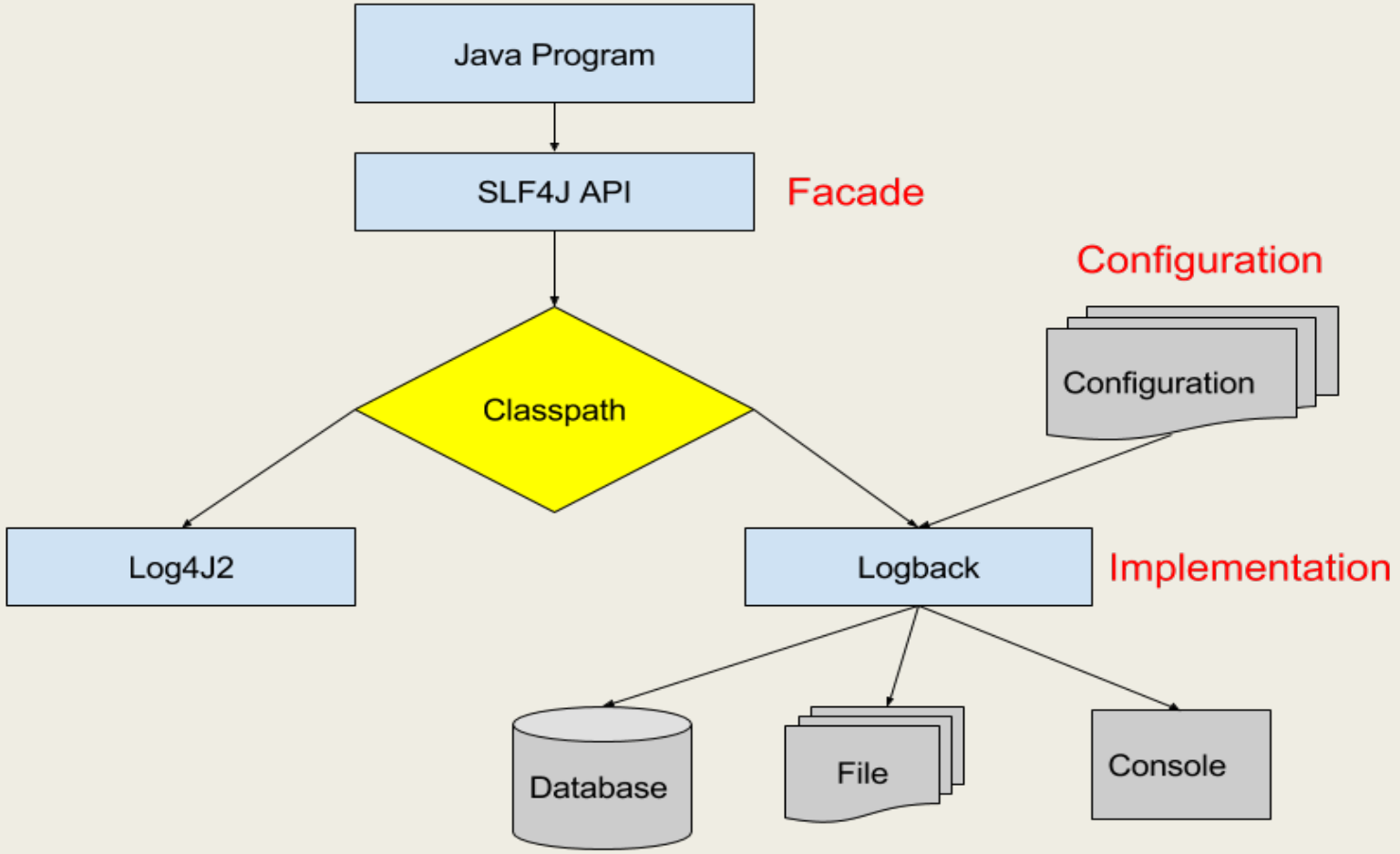
Architecture I

- Modern logging = Facade + Implementation + Configuration + Aggregator

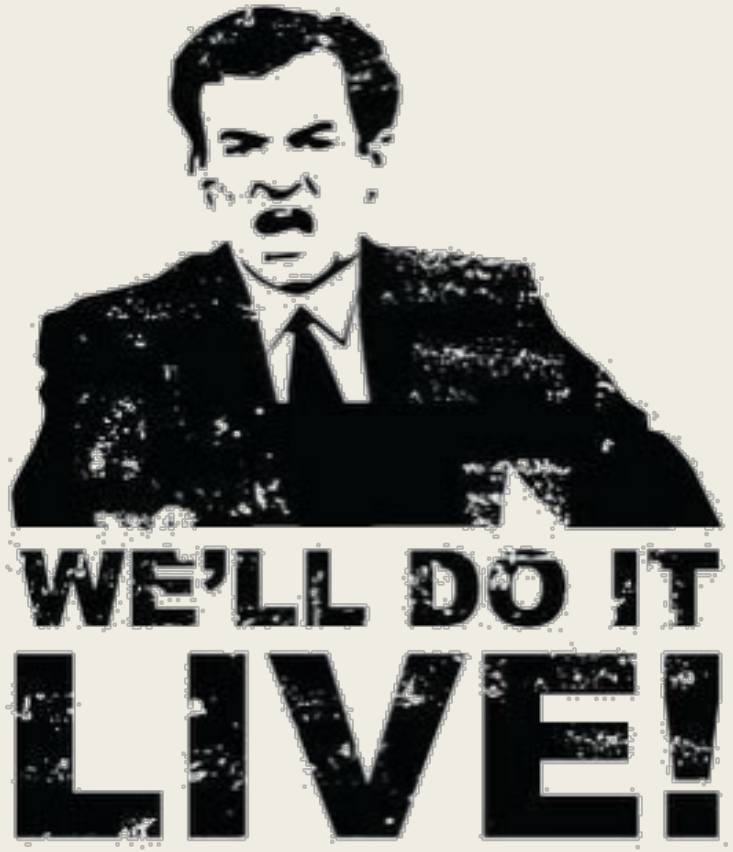
Architecture II

- Modern logging = Facade + Implementation + Configuration + Aggregator
- Let’s consider only the first 3 for now.

Architecture III



Examples



Examples - Setup

1. Clone from Github: https://github.com/istvan-fodor/logging-examples
    1. *If you look at `TestLogging.java`, you will see that logging features are demonstrated in individual JUnit4 test methods.*
2. Run `src/test/resources/run-docker.bat` (If you don't have Docker installed, get it here.)
3. Run "`mvnw clean test`" to see if everything works. You should see "BUILD SUCCESS".
4. We will go over individual test cases.

Examples – How and what I

- To run individual tests from command line:
  - *`mvnw clean test -Dtest=TestLogging#<test name>`*
- Tests:
  - *testLevels: shows different logging levels and when they get displayed*
    - Tip: Change the level on `<logger name="com.ifodor" level="INFO" />` to `TRACE`
  - *testBigObject: shows why you should use substitution instead of concatenation and why you should avoid `toString()` in log message parameters*
  - *testMdc: demonstrates Mapped Diagnostic Contexts. Observe the log messages (callerId = ?)*

Examples – How and what II

- Tests:
  - *testMdcFile: demonstrates file logging into separate files based on MDC values*
    - Tip: check the target/logs folder. Uncomment the Stream and run the test again. Check the `target/logs/archive` folder too!
  - *testDatabase: demonstrates the DBAppender (logging to a database! .. Its in the computer!)*
    - Run the Postgres Docker container (README.md), connect with root/password to localhost. Run `select * from logging_event`.
    - Tip: we are logging to HSQLDB too! Open `target/db/logs.script`, see the inserts on the bottom.
    - Tip: See how we initialized the 2 databases with DDL scripts (`init()`)

Resources

- https://logback.qos.ch
- https://www.slf4j.org
- https://logging.apache.org/log4j/2.x/
- Log aggregation: https://www.elastic.co/webinars/introduction-elk-stack