Loggery

Marcin Kwiatkowski

Logi

Komunikaty diagnostyczne z działania aplikacji

Logi - podstawowe informacje

- Pozwalają namierzyć problem, który wystąpił w systemie (po awarii lub w jej momencie)
- Programista jest odpowiedzialny za dodawanie logowania
- **Mechanizmy tworzące logi nazywamy loggerami**

```
 9
10          logger.info("This is an info message");
11
12          logger.warn("This is a warning!");
13
14          logger.error("Error!");
15
```

Co należy logować?

- Kolejne kroki w ważnych procesach / algorytmach aplikacji
- Obsłużone wyjątki (jeśli wykonany zostanie kod w sekcji catch)
- Informacje o użytych konfiguracjach aplikacji (np. stage, CI lub production)

Cechy logów

- Logger przekazuje logi do określonego miejsca docelowego (np. standardowe wyjście, plik lub serwer logów)
- Każdy pojedynczy log ma:
  - Poziom logowania
  - Czas zarejestrowania
  - Treść

Poziom logowania 1

- Cecha pojedynczego loga (komunikatu)
- Określenie ważności loga
- Biblioteki dostarczające implementacje loggerów mogą mieć różne poziomy logowania, np.
  - log4j - **FATAL, ERROR, WARN, INFO, DEBUG, TRACE**
  - Java Logging API - **SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST**
- Można definiować własne poziomy

Poziom logowania 2

Poziom	Waga
OFF	0
FATAL	100
ERROR	200
WARN	300
INFO	400
DEBUG	500
TRACE	600
ALL	Integer.MAX_VALUE

Zastanów się...

do czego może przydać się poziom logowania?

Kilka porad z doświadczenia...

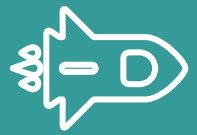
- Zachować zdrowy rozsądek przy tworzeniu logów.
- Dodając logi zastanów się, które wypisane informacje mogą się przydać gdy system ulegnie awarii.
- Nie zostawiaj pustych sekcji catch
- Nie pisz loggerów samemu! Loggery muszą być napisane w taki sposób, aby nie blokować i nie spowalniać standardowego działania aplikacji

Biblioteki z loggerami

- **log4j2**
- **Java Logging API** (java.util.logging) - wbudowane w JRE
- **slf4j**
  Uwaga! To jest tylko API, bez implementacji.

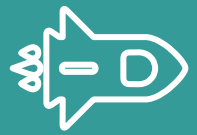
- Logback - https://logback.qos.ch

Log4j2



- Strona główna - https://logging.apache.org/log4j/2.x
- Jeden z bardziej znanych i używanych loggerów

Zadanie 1



- Zaimportuj projekt `01-loglevel`
- Zaobserwuj jak zmieniają się wypisywane logi w zależności od ustawionego poziomu logowania

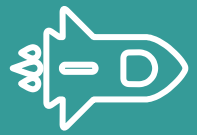
Log4j2 - konfiguracja

- Log4j2 szuka ścieżki do pliku konfiguracyjnego w zmiennej systemowej `log4j.configurationFile`
  - `System.setProperty(„log4j.configurationFile”, „log4j.xml”);`
  - `-Dlog4j.configurationFile=log4j.xml`
- Domyślny poziom ERROR, wypisywanie na konsolę

Log4j2 - appenders

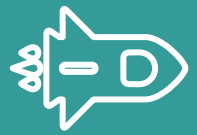
- **Appender** jest konfiguracją miejsca docelowego do składowania logów.
- Domyślnie logi trafiają na standardowe wyjście (na konsolę).
- Można ustawić, aby logi były wypisywane do pliku.
- Zapisywanie logów do pliku jest domyślnie **synchroniczne**.
  Co to oznacza w praktyce?
- Można tryb synchroniczny zamienić na asynchroniczny.

Zadanie 2



- Zaimportuj projekt `02-file-appender`
- Uruchom aplikację tak, aby logi zostały wypisany do pliku w sposób synchroniczny oraz asynchroniczny.

java.util.logging

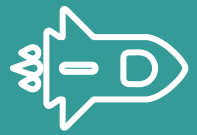


- Standard języka Java, wchodzi w skład JRE
- Mniejsze możliwości konfiguracyjne niż log4j2

java.util.logging - konfiguracja

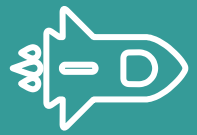
- Najniższy dopuszczalny poziom logowania jest ustawiany w ustawieniach JRE
- Można zmieniać poziom logowania w aplikacji poprzez `logger.setLevel(Level)`
- Dopóki poziom nie zostanie ustawiony, zawsze jest brany z „rodzica” w hierarchii

Zadanie 3



- Zaimportuj projekt `03-prime-test`
- Dodaj do projektu logi, dzięki którym będzie można prześledzić działanie aplikacji po jej zakończeniu działania
- Korzystając z polecenia logger.setLevel(Level) zobacz jak działa ustawianie poziomu dla java.util.logging

SLF4J



- Strona główna - https://www.slf4j.org/
- Warstwa abstrakcji

SLF4J - API, interfejs

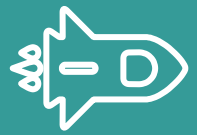
- SLF4J to **NIE** jest implementacja loggera tak jak np. log4j2.
- SLF4J jest zbiorem interfejsów, które umożliwiają zapisywanie logów
- SLF4J wymaga obecności biblioteki implementującej logowanie
- SLF4J jest m.in. implementowane przez:
  - log4j, log4j2
  - java.util.logging
  - Logback

SLF4J - inicjalizacja

- Inicjalizacja SLF4J jest nieco inna niż dla log4j:

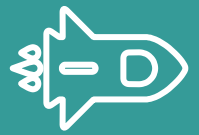
```
Logger logger = LoggerFactory.getLogger(App.class.getName());
logger.info("This is an info message");
```

Zadanie 4



- Zaimportuj projekt `04-loglevel-slf`
- Zmień logowanie z log4j2 na SLF4J.
- Podepnij log4j2 jako implementację SLF4J.
- Zmień log4j2 na java.util.logging

Zadanie 5



- Wraz z zespołem uruchom SLF4J wraz z log4j w swoim projekcie.

Dzięki

LinkedIn: https://bit.ly/2Bzapg3

- Trener infoShare Academy
- Full Stack Developer w Bright Inventions