Java Coding Guidelines - Visual Board GmbH

Dokumentenmanagement

Erstellungsdatum:	22.06.2020
Autoren:	Oliver Egloff (OEG)
Dateiname:	OEG-LWI-SSA_Backend_Coding_Guidelines_0.1_Draft_4

Änderungsverzeichnis

Version	Datum	Autor	Beschreibung
1.0	22.06.2020	OEG	Coding Guidelines erstellt

Inhaltsverzeichnis

1 Code Style **3**

2 Logging **3**

3 Validierung **3**

4 Variablenbennenung **3**

5 Verwendung von primitives und object types **3**

6 Spring Boot **3**
6.1 Bean Deklaration 3
6.2 POJO Namensgebung - DAO und DTO 4
6.3 Config-Objekte 4
6.4 lombok 4

7 Tests **4**

1 Code Style

In unseren Java Projekten verwenden wir den Code Styleguide von Google. Dieser kann für IntelliJ hier heruntergeladen werden https://github.com/google/styleguide/blob/gh-pages/intellij-java-google-style.xml
Den Styleguide kann in IntelliJ wie folgt eingerichtet werden: https://www.jetbrains.com/help/idea/configuring-code-style.html

2 Logging

Abweichung zu Konstanten-Style → Klein- und Kurzschreibung

```
private final static Logger log = new Logger(....class);
```

3 Validierung

Für die Validierung sind folgende 2 Konstrukte zulässig:
boolean is Valid / **has**Valid...(Email)
void validate... / **check**...(Email) **throws** ...Exception

4 Variablenbennenung

Für die Validierung sind folgende 2 Konstrukte zulässig:

Regel	**Beispiel**
Programmieren immer auf Englisch	
auch bei Variablennamen	spielerName → nickname
ganze Wörter	pId → playerId
camelCase	playername → playerName

Ausnahme: “Exception e“ in einem Catch-Block

5 Verwendung von primitives und object types

- Wenn möglich immer primitives verwenden.

  Für Flags / boolische Werte immer den primitiven Datentyp anwenden

- Kommen boolean values aus einer Datenbank und können auf dieser Ebene nicht sauber true / false zugeordnet werden, so gilt:

  Wenn Wert == null → false, sonst Wert = DB value

6 Spring Boot

6.1 Bean Deklaration

- services classes → **@Service(“AUTO GENERATED UUID“)** → um verschiedene Versionen zu unterscheiden
- controller → **@RestController(“AUTO GENERATED UUID“)** → um verschiedene Versionen zu unterscheiden

- alle Klassen, welche die als Bean für dependency injection verfügbar sein sollen
  → **@Component(“AUTO GENERATED UUID“)**
- BeanConfigurations (Klassen mit Bean Factory Methoden)
  → @Configuration (https://stackoverflow.com/a/39175018)

6.2 POJO Namensgebung - DAO und DTO

Objekte (POJOs) werden für den Datenaustausch in der Applikation und zwischen Client und Server immer **DTO** verwendet.
Mapped ein Objekt eine Datenbank Tabelle, dann soll dafür ein **DAO** verwendet werden. Soll ein Datenbank Objekt dem Client zur Verfügung gestellt werden, dann muss es zuerst mithilfe einer Mapper-Klasse auf ein DTO gemapped werden.

POJO Namensgebung	**Beispiel**
DAO	SomeObject...**DAO**
DTO	SomeObject...**DTO**

6.3 Config-Objekte

Config Objekte für die *application.yml* Dateien, müssem mit @Component und @ConfigurationProperties(“CONFIG PREFIX“)
In Config Objekten werden die Getter und Setter NICHT durch lombok generiert!

6.4 lombok

Das “Lombok” Plugin kann hier heruntergeladen werden: https://plugins.jetbrains.com/plugin/6317-lombok
Generell nur für DTOs und DAOs verwenden. In Config-Objekten immer eigene Getter und Setter erzeugen.

7 Tests

- Für Tests verwenden wir JUnit 5 (Jupiter)
- Mock Objekte werden mit dem Präfix **mock**SomeObject
- Test Method Naming

  Der Name einer Test-Methode zeigt immer auf, was genau getestet wird

  Für die Schreibweise wird camelCase oder mixedCamel_withSnakeCase verwendet

  Beispiel Test Methoden Namen

  testGetCustomerByIdTrowsExceptionOnNull

  doesFetchData

  buildSomeObjectIsSuccessful