

## Arbeitsblatt 9: Validierung

### Ziele

Sie haben für das Create Formular eine funktionierende Validierung der Eingabewerte implementiert.

### Ausgangslage

Sie haben die Übung 4 korrekt gelöst und sie können das Projekt erfolgreich ausführen. Auf dem AD finden sie eine mögliche Lösung unter "Lektion04/AB7-AB8-UB4-solution".

### Aufgabe 1: Validierung auf Datenbankschicht einführen

Forcieren sie eine Validation Exception auf der Datenbankschicht. Ergänzen sie dazu die Property "title" und die Property "description" der Entität `Questionnaire` mit der JSR-303 Annotation `@Size`, so dass der Titel 2-30 Zeichen und die Beschreibung 10-50 Zeichen umfassen muss.

Zusätzlich ergänzen die Klasse `FlashcardApplication` mit folgenden Spring Beans:

```
@Bean
public ValidatingMongoEventListener validatingMongoEventListener() {
    return new ValidatingMongoEventListener(validator());
}

@Bean
public LocalValidatorFactoryBean validator() {
    return new LocalValidatorFactoryBean();
}
```

Starten sie die flashcard-Applikation und versuchen sie eine neue `Questionnaire` Entität anzulegen, deren Titel nur einen Buchstaben enthält.

Analysieren sie die Exception.

### Aufgabe 2: Validierung im Controller einführen

Es ist sinnvoll die Validierung auf der Serverseite möglichst früh durchführen zu können, so dass keine unnötigen Ressourcen verbraucht werden und der Benutzer schnell ein Feedback erhält. Deshalb ist die Validierung beim Controller zweckmässig.

Um auf einer Entität, wie in Aufgabe 1 eingeführt, die Validierung zu forcieren, stellt JSR-303 die `@Valid` Annotation zur Verfügung. Das Spring Framework kann nun diese Annotation in einer Controller-Methode erkennen und eine Validierung durchführen. Das Framework wird das Resultat der Validierung in eine Instanz vom Typ `BindingResult` ablegen. Über diese Instanz kann man anschliessend auf Validierungsfehler prüfen und je nach Resultat dann die entsprechende Response auslösen.

**Wichtig:** Die `@Valid` Annotation muss vor der entsprechenden Entität stehen.

### Tipps:

- Nutzen sie die Informationen unter <https://spring.io/guides/gs/validating-form-input/>
- Die Beans aus Aufgabe 1 können wieder gelöscht werden.

Implementieren sie in der Controller-Methode `QuestionnaireController.create()` folgende Struktur:

```
public String create(@Valid Questionnaire questionnaire,
                    BindingResult result) {
    if (result.hasErrors()) {
        // return which view?
    }
    ...
    // return which view?
}
```

Was muss der Controller als logischen View Namen zurückgeben:

1. im Fehlerfall
2. nach einer erfolgreichen Validierung und Speicherung der neuen Entität

### Aufgabe 3: Validierungsfehler anzeigen

Die Validierung ist erst sinnvoll wenn auch der Benutzer über fehlerhafte Eingaben in einer ansprechenden und übersichtlichen Art informiert wird. Deshalb braucht es auf der View-Schicht entsprechende Mechanismen. Die gute Integration von Thymeleaf in SpringMVC macht aber auch diese Aufgabe zu einem Kinderspiel.

Erweitern sie das Formular so, dass fehlerhafte Eingaben "benutzerfreundlich" angezeigt werden (siehe Abb. 1).

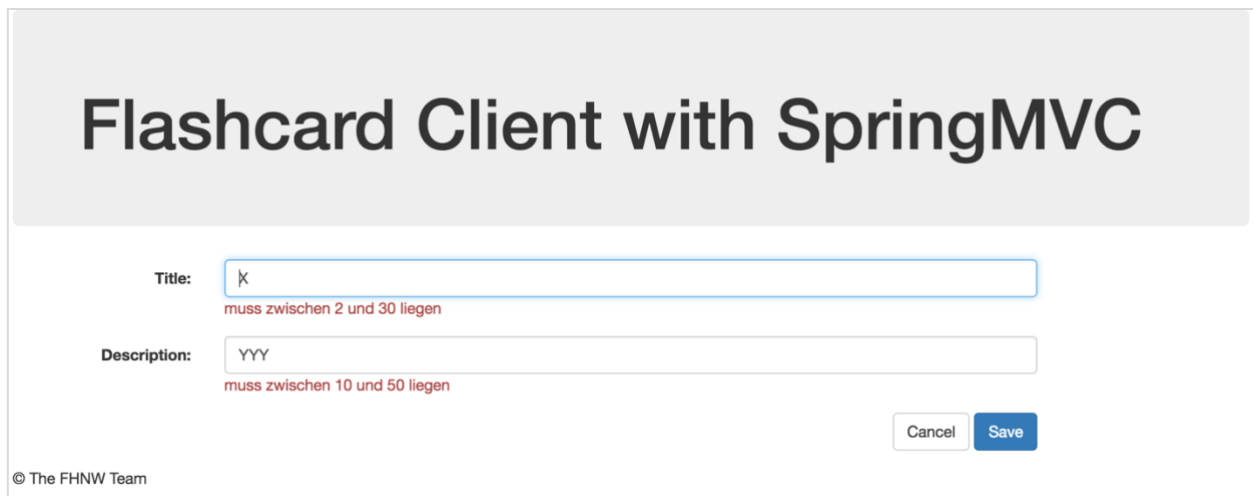


Abb. 1: Fehlermeldungen bei ungültigen Formulardaten

#### Tipps:

- Nutzen sie die Informationen unter <https://spring.io/guides/gs/validating-form-input/>
- Die sprachabhängigen Fehlermeldungen sind im File "ValidationMessages\_de.properties" in der JAR-Bibliothek hibernate-validator-XXX.jar zu finden.

Dabei ist es wichtig, dass nach einer fehlerhaften Validierung sofort wieder zum Formular gewechselt wird, wobei die alten Werte immer noch sichtbar sind.

Für das Einfärben der Fehlermeldung in Rot müssen sie ein neues CCS-File erstellen und dieses File in die Applikation integrieren. Lesen mehr dazu im Kapitel "27.1.5 Static Content" der Spring Boot Reference Dokumentation.