295 - Backend

Auftrag Fallstudie 6 - Validation

|  |  |
| --- | --- |
| Richtzeit  30 Minuten | Sozialform  Einzelarbeit |

## Beschreibung

Damit wir überprüfen können, ob die vom Client gesendeten JSON-Objekte auch dem entsprechen, was wir erwarten, müssen wir diese validieren.

## Auftrag

Verwende folgende Annotationen auf den Feldern, die validiert werden müssen.

|  |  |
| --- | --- |
| Annotation | Beschreibung |
| @NotBlank | Überprüft, dass der folgende String nicht leer ist. Beinhaltet ein @NotNull. |
| @NotNull | Überprüft, dass der folgende Wert nicht null ist. |
| @Size(min = 6, max = 255) | Überprüft, dass der folgende String eine Länge von mindestens 6 und maximal 255 hat. (Kann auch für listen verwendet werden) |

Weitere können in der [offiziellen Doku von javax.validation.constraints](https://docs.oracle.com/javaee/7/api/javax/validation/constraints/package-summary.html) gefunden werden.

Versuche nun, das ItemRequestDTO sowie das TagRequestDTO mit entsprechenden Annotationen zu versehen. Als Vorlage steht dir die Person-Klasse zur Verfügung. Gib der Annotation als message eine sinnvolle Nachricht mit, falls die Validierung scheitert.

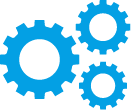
Damit die Validierung auch durchgeführt wird müssen die zu validierenden Parameter der Controller-Methoden mit der Annotation @Valid annotiert werden. Als Beispiel hier die signUp-Methode zum Erstellen einer Person.

public ResponseEntity<?> signUp(@Valid @RequestBody AuthRequestDTO authRequestDTO)

Da bei PATCH nicht immer alle Felder existieren, müssen wir beim update das @Valid weglassen.

**Teste deinen Code**

Führe alle automatischen Tests durch und teste auch manuell, ob die erwarteten Fehler geworfen werden, solltest du z.B. für ein Item keinen Namen mitgeben.



### Manuelle Validierung bei PATCH

Beim Updaten einer Entität wollen wir nicht das ganze Objekt validieren, da, wie oben schon erwähnt, nicht alle Felder existieren müssen. Wir lassen also das @Valid weg. Somit haben wir keine Überprüfung, ausser ob ein Feld null ist. Wenn wir jedoch Felder haben, die beispielsweise nicht leer sein dürfen, wie der Name des Items müssen wir überprüfen, ob dies wirklich so ist. Es kann sein, dass ein Leerer String mitgegeben wird.

Wir müssen also die merge-Methode im Service noch einmal anpassen. Stand jetzt sollte geprüft werden, ob ein Feld nicht null ist und wenn dies zutrifft, wird es direkt so übernommen. Nun wollen wir bei den Feldern, die im DTO bereits annotiert sind (@NotBlank, @Lenght usw.) beim mergen manuell prüfen, ob sie valid sind.

Erstelle am Anfang der merge-Methode eine Map mit einem String als Key und einer Liste von Strings als Value.

Map<String, List<String>> errors = new HashMap<>();

Die Überprüfung, ob ein Feld Null ist, behalten wir so bei, prüfen aber bei den zu validierenden Feldern, ob sie dem entsprechen, was wir wollen. Nehmen wir als Beispiel den Username der Person. Dieser darf nicht leer sein. Nachdem der null-Check durch ist, überprüfen wir mit StringUtils, ob auch etwas darinsteht.

if (changing.getUsername() != null) {  
 if (StringUtils.*isNotBlank*(changing.getUsername())) {  
 existing.setUsername(changing.getUsername());  
 } else {  
 errors.put("username", List.*of*("username must not be empty"));  
 }  
}

Falls nichts darinsteht oder nur Leerzeichen fügen wir der Map errors einen neuen Eintrag hinzu. Als Key nehmen wir den Namen des Feldes (username) und als Value eine sinnvolle Nachricht. Um aus einem String einfach eine Liste zu machen kannst du List.of() verwenden. Dies wiederholen wir für alle Felder, die geprüft werden sollen.

Am Schluss der merge-Methode, falls errors nicht leer ist muss eine Exception geworfen werden. Erstell dafür die Klasse FaildValidationException im Package ch.bbcag.backend.todolist.  
Diese Erbt von RuntimeException und hat eine Instanzvariable errors mit dem Typ  
Map<String, List<String>>. Die Exception werden wir zu einem späteren Zeitpunkt abfangen und behandeln.

if (!errors.isEmpty()) {  
 throw new FailedValidationException(errors);  
}

Mach diese Änderungen bei allen Services (TagService, ItemService und PersonService).