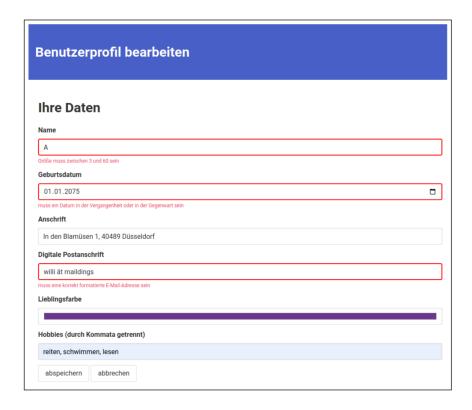
Webbasierte Anwendungen

04.05.2022

Aufgabe 1 (Validierung)

Momentan kann man noch eine Menge Unsinn in unserem Benutzerprofil-Editor erfassen – aber nicht mehr lange:

- Fügen Sie bitte den dependencies in Ihrer build.gradle-Datei die neue Abhängigkeit org.springframework.boot:spring-boot-starter-validation hinzu, da hier die nötigen Annotationen und Validierungsfunktionalitäten bereitgestellt werden. Vergessen Sie nicht, wie immer nach Änderungen an build.gradle per Kontextmenü im VSCode ein "Update Project" anzustoßen.
- Bitte nutzen Sie die Standard-Annotationen der "Bean Validation", um in BenutzerProfil zu hinterlegen, ein Geburtsdatum aus der Zukunft nicht statthaft ist.
- Es soll zudem deklarativ sichergestellt werden, dass der **Name** mindestens drei und höchstens 60 Zeichen lang ist.
- Natürlich soll das email-Attribut eine gültige E-Mail-Adresse enthalten. Die zugehörige Standard-Annotation lässt auch null zu, was hier ok ist. Daher ändern wir auch die Initialisierung von email vom Leerstring auf null.
- Die Attribute name, geburtsdatum, adresse und interessen dürfen nicht mit null belegt werden.
- Bitte erweitern Sie Ihren Controller (sofern noch nicht geschehen) geeignet, damit Validierungsfehler beim POSTen des Bearbeitungsformulars erkannt und die Fehler-Hinweise in einem BindingResult aufgefangen werden. Wenn beim Abschicken (POST) des Bearbeitungsformulars ein Validierungsfehler auffällt, soll der Benutzer auf dem (mit den vorherigen Eingaben vorausgefüllten) Bearbeitungsformular bleiben (einfach gleiche View renderen, kein redirect zur Übersichtsliste), damit er die Fehler sehen, sich schämen und die Fehler korrigieren kann.
- Ergänzen Sie das Bearbeitungsformular nun so, dass die **Fehlermeldungen** beim zugehörigen Eingabefeld erscheinen (die unten gezeigten Meldungen sind mitgelieferte Standard-Texte des Validierers, kein Zusatzaufwand erforderlich).
- Zudem sollen (nur) die fehlerbehafteten Eingabefelder mit einem roten Rahmen umkringelt werden (CSS "border", 2px breit, rot).



Aufgabe 2 (Eigener Validator)

Momentan kann man in das "Lieblingsfarbe"-Eingabefeld noch jeden Unsinn eingeben (es ist ein String-Feld – nur der Browser und seine Darstellung von INPUT-Feldern vom Typ color per Color-Picker stehen zwischen uns und dem chromatischen Chaos, es gibt aber auch Browser, die hier nur ein Texteingabefeld zeigen). Das zeigt, dass die Serverseite auch diese Eingabe checken muss.

Für das Attribut lieblingsfarbe von BenutzerProfil sollen nur RGB-Werte in der bekannten Form mit führendem "#", gefolgt von entweder drei (z.B. #1d4) oder sechs (z.B. #496BC7) hexadezimalen Ziffern erlaubt sein, wobei Groß-/Kleinschreibung der Hex-Ziffern a bis f erlaubt ist.

Zudem haben Amateurpsychologen herausgefunden, dass Farben eher als *bunt* wahrgenommen werden, wenn die R-, G- und B-Anteile der Farbe alle paarweise unterschiedliche Werte haben. Beispiele für bunte Farben wären z.B. #123 oder #albcla, Beispiele für unbunte Farben wären also ein solides Mittelgrau #aaa, ein Schlumpfgrün #10Cf10 oder ein edles Frikadellenblau #171742.

Bitte legen Sie im Package validierung eine Annotation @Bunt für String-Variablen an, zusammen mit einer Klasse BuntValidator, welche die Einhaltung der Restriktion sicherstellt.

Gültige Eingabewerte sind null, der Leerstring und ein RGB-Wert in einem der beiden o.g. Formate, wobei (unter Ignorieren von Groß-/Kleinschreibung bei den Hexziffern) alle Werte der drei Farbkomponenten unterschiedlich sein müssen. Anderenfalls ist der Eingabewert ungültig.

Fügen Sie Ihre @Bunt Annotation danach dem lieblingsfarbe-Feld Ihrer BenutzerProfil-Klasse hinzu und testen Sie, ob unbunte Lieblingsfarben nun in Ihrer Webanwendung ordnungsgemäß abgewiesen werden.



Aufgabe 3 (I18n und L10n)

"Internationalisierung" scheint ja so wichtig zu sein, dass es in unserem Curriculum sogar ein eigenes Modul dafür gibt.

überarbeiten Sie mindestens Beschriftungen Bitte die der Benutzerprofilansicht (profilansicht.html) und die der Eingabefelder des Bearbeitungsformulars (profileditor.html) Ihrer Anwendung so, dass die Textschnipsel für die Überschriften bzw. die Formular-Labels nach Möglichkeit in der vorgewählten Sprache erscheinen, abhängig von der im Browser eingestellten Sprach-Präferenz (und als Fallback auf Deutsch). Bitte testen Sie, ob Ihre Webanwendung auf Änderungen der Browser-Spracheinstellung wie gewünscht reagiert. Legen Sie dazu bitte (mindestens) für Deutsch und Englisch die passenden Übersetzungs-Dateien an und referenzieren Sie deren Einträge in Ihren Thymeleaf-Templates. Das Datumsformat des Geburtsdatums braucht *nicht* umgestellt zu werden – es geht hier nur um die Texte.

Wegen der Unit-Tests sollte man sich auf die Benennungen einigen, wir nehmen folgende, wobei Groß-/Kleinschreibung in den Tests egal ist. Sie finden die Texte auch als leicht bearbeitbare CSV-Textdatei unter dem Namen ueb04-uebersetzungen.csv im read.MI. Diese enthält eine zusätzliche Spalte mit Property-Namen, so dass man sich mit wenigen Zeilen Python ein Skript erstellen kann, das die drei message_xx.properties-Dateien aus dem CSV-File generiert. So kann man die Übersetzungen z.B. mit Excel oder LibreOffice Calc übersichtlich pflegen und bekommt stets konsistente properties-Dateien für beliebig viele Sprachen per Knopfdruck aus dem Skript, die dann nur noch in das Projekt kopiert werden müssen. Und wer kein Python kann, macht's halt von Hand (sollte aber in sich gehen).

de	en	nl
Ihre Daten	Your data	Uw gegevens
Name	Name	Naam
Geburtsdatum	Date of birth	Geboortedatum
Anschrift	Street address	Adres
Digitale Postanschrift	EMail address	Digitaal postadres
Lieblingsfarbe	Favourite colour	Favoriete kleur
Hobbies	Hobbies	Hobby's
abspeichern	save	opslaan
abbrechen	cancel	anuleren
bearbeiten	edit	bewerken
Benutzerprofil	User profile	Gebruikersprofiel
Benutzerprofil bearbeiten	Edit user profile	Gebruikersprofiel wijzigen
Hobbies (durch Kommata getrennt)	Hobbies (comma-separated list)	Hobbies (gescheiden door komma's)

Denken Sie bitte auch daran, die Meldung bei Verstößen gegen Ihre @Bunt-Restriktion auf Deutsch und Englisch anzubieten (https://deepl.com hilft beim Übersetzen). In die Übersetzung soll der validierte Wert eingebettet werden, damit der Nutzer seinen Fehler (zwei gleiche R/G/B-Farbwerte) besser erkennt z.B. so:

Lieblingsfarbe
Diese Farbe #5c2020 enthält zwei gleiche R/G/B-Anteile und ist daher nicht bunt genug



Sobald die Anwendung einmal durch den Umbau *internationalisiert* ist, kann sie ohne Codeänderung jederzeit nur durch Hinzufügen weiterer messages-Dateien für neue Sprachen *lokalisiert* werden.

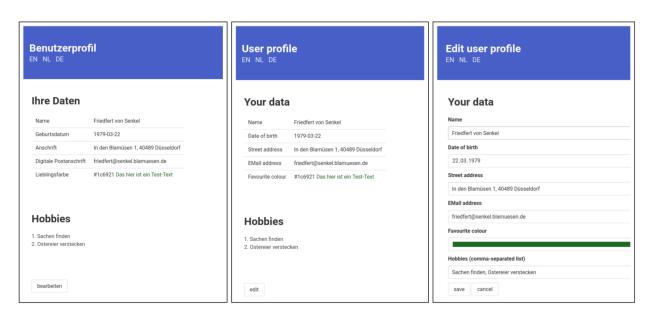
Aufgabe 4 (Language-Switch)

Versehen Sie bitte nun zum Abschluss Ihre Anwendung noch mit einem Umschalter der Sprach-Präferenz innerhalb der Anwendung, damit der Benutzer nicht immer an seinen Browsereinstellungen herumspielen muss.

Fügen Sie dazu bitte passende Links im Kopf Ihrer Profilansicht- (profilansicht.html) und der Bearbeitungs-Seite (profileditor.html) hinzu, um per Query-Parameter sprache entweder Deutsch (de) oder Englisch (en) (und natürlich gerne weitere) auswählen zu können.



Erstellen Sie bitte im Package configuration eine geeignete @Configuration-Klasse, um den Query-Parameter sprache abzufangen und die gewählte Sprache zu setzen. Testen Sie nun, ob Sie mit Hilfe Ihrer Sprach-Links die Anwendungssprache umstellen können (und ob dies auch beim Wechsel von der Profilansicht auf die Bearbeitungs-Seite (und zurück) erhalten bleibt).



Hinweis: Wenn Sie einen HTML-Link href="?param=wert" bzw. in Thymeleaf mit @ {th:href="?param=wert"} anlegen, wird ein GET-Request auf dieselbe Seite, aber mit dem/den angegebenen Query-Parametern ausgeführt. Sie müssen also nicht den URI-Pfad komplett hardcoden und können denselben HTML-Schnipsel für die Sprachumschaltung in beiden Templates verwenden.