

Webbasierte Anwendungen

Web-Ueb-03 (06.05.2024)

Aufgabe 1 (Formular-Validierung)

Momentan werden die Eingaben in unserem Benutzer-Formular noch nicht überprüft, so dass es möglich ist, z.B. un plausible Geburtsdaten anzugeben oder wichtige Felder leer zu lassen. Das ist nicht gut.

- 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 `BenutzerFormular` zu hinterlegen, dass das Namens-Eingabefeld nicht leer gelassen werden darf und zwischen jeweils einschließlich 3 und 80 enthalten muss.
- Das **Geburtsdatum** muss in der Vergangenheit liegen
- Die **E-Mail-Adresse** muss zumindest syntaktisch eine halbwegs plausible E-Mail-Adresse sein (es gibt einen Standardvalidierer dafür). Wenn Sie im HTML-Formular dem Eingabefeld den Typ `email` geben, kann es sein, dass der Browser selbst fehlerhafte Mailadress-Eingaben abfängt. Da wir aber nicht wissen, ob und wie er das tut, ist es *wichtig*, die Eingabe-Validität *auf der Serverseite* sicherzustellen. HTTP-Requests lassen sich einfach manipulieren – Browser-seitige Validierung ist daher **immer** nur ein Komfort-Feature, sich darauf verlassen und auf serverseitige Validierung verzichten darf man *nie*. Bitte testen Sie daher, ob Ihre serverseitige Validierung funktioniert, indem Sie dem E-Mail-Eingabefeld den Typ “**text**” geben, um eine eventuelle Browser-seitige Vorvalidierung zu verhindern (zu Lernzwecken natürlich – ansonsten spricht nichts gegen treffende INPUT-Typen).
- 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, damit er die Fehler sehen, sich schämen und die Fehler korrigieren kann.

Ein **Sonderfall** ist das Passwort-Feld, dieses wird standardmäßig nicht wieder mit einer vorherigen Eingabe befüllt (sonst könnte jeder das Passwort im HTML-Quelltext in seinem Browser nachlesen – Sie sollten dafür nichts tun müssen, wenn das Eingabefeld als vom Typ `password` angelegt wurde. Wenn Sie den Eingabefeld-Typ auf `text` ändern, sollten Sie eine vorherige Eingabe dagegen wieder sehen – probieren Sie es bitte mal aus).

- 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).

Hier ein Beispiel (“Beispiel” bedeutet, dass da nicht alle Fehlerfälle auf einmal enthalten sein können. Ein Beispiel ist **keine** vollständige Spezifikation. Wenn etwas nur für ein paar Beispiele geht, sagt das nur, dass es zumindest dafür (aber nicht allgemein) funktioniert – zum Entwickeln zählen daher **immer** die Gesamtheit der formulierten Validierungsbedingungen im Text). Sie sehen z.B. hier eine browser-seitige Prüfung des E-Mail -Felds, und das sollte bei Ihnen ja *nicht* auftreten, weil Sie zu Testzwecken den Eingabefeld-Typ auf “text” gesetzt haben, weswegen Ihre Mail-Validierungsfehlermeldungen wie bei den anderen beiden validierten Feldern aussehen – oder?

Benutzerprofil 17 bearbeiten

Name
Oh
Größe muss zwischen 3 und 80 sein

E-Mail
meinefalscheadresse
Die E-Mail-Adresse muss ein @-Zeichen enthalten. In der Angabe "meinefalscheadresse" fehlt ein @-Zeichen.

Passwort
Passwort

Geburtstag
31.12.2030
muss ein Datum in der Vergangenheit sein

Ich mag (max. 5)
Ich mag

Ich mag nicht (max. 5)
Ich mag nicht

ok

Aufgabe 2 (Eigener Validator)

Sichere Passwörter sind für unser System von entscheidender Bedeutung. Da die Webanwendung potentiell Milliarden interessierten Nutzern zur Verfügung steht, müssen Passwörter bestimmten Mindestanforderungen genügen, um nicht einfach erraten werden zu können.

Bitte implementieren Sie im Package `validators` ein Validator mit `@GutesPasswort`-Annotation, die auf einen `String` angewandt werden kann und feststellt, ob der zu validierende String entweder die Ziffernfolge “17” oder das Wort “siebzehn” enthält. Groß-/Kleinschreibung ist dabei egal, zulässig wäre also auch z.B. “SiEbzeHN” oder “siebZEHN”.

Wenn ein `null`-Wert oder ein Leerstring übergeben wird, sei das per Definition valide (vgl. Kommentar oben zu `password`-Eingabefeldern).

Fügen Sie Ihre `@GutesPasswort` Annotation bitte danach der `password`-Feld Ihrer `BenutzerFormular`-Klasse hinzu und testen Sie, ob im obigen Sinne unsichere Passwörter korrekt abgewiesen werden.

Benutzerprofil 17 bearbeiten

Name

E-Mail

Passwort

Das ist kein gutes Passwort

Geburtstag

Ich mag

(max. 5)

Ich mag nicht

(max. 5)

Hinweis: Die Validierung erfolgt beim serverseitigen *Empfang* des Formularinhalts. Validierungsfehler werden also erst bei der erneuten Anzeige des Formulars im Rahmen einer “Korrekturschleife” ausgegeben, nicht schon während der Eingabe.

Aufgabe 3 (I18n und L10n)

Nun wollen wir unsere Anwendung auch mehrsprachig anbieten.

Bitte überarbeiten Sie die Beschriftungen des `benutzerbearbeiten`-HTML-Formulars so, dass die Textschnipsel für die Überschriften und 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. Wir benötigen mindestens für die Sprachen Deutsch, Englisch und Niederländisch die passenden Übersetzungs-Dateien und referenzieren deren Einträge im Thymeleaf-Template.

Im `read.MI` finden Sie eine CSV-Datei `uebersetzungen.csv`, welche mögliche Propertynamen für Übersetzungen enthält, dazu Textvorschläge für Deutsch (`de`), Englisch (`en`) und Niederländisch (`nl`).

Die erste Zeile der Semikolon-getrennten CSV-Datei besteht aus dem Eintrag `propertyname`, gefolgt von einer (beliebig langen) Folge von korrekten Sprachkürzeln, die Sie so in den Dateien `messages_xx.properties` übernehmen können. Die nachfolgenden Übersetzungs-Zeilen sind analog aufgebaut (Propertyname, gefolgt von Textschnipseln je Sprach-Spalte).

```

propertyname;de;en;nl
# Kommentarzeile
benutzer.name;Name;Name;Naam
benutzer.email;E-Mail;E-mail;E-mail

benutzer.passwort;Passwort;Password;Wachtwoord
benutzer.geburtstag;Geburtstag;Date of birth;Geboortedatum
...

```

Bitte kopieren Sie die Datei auf der obersten Verzeichnisebene in Ihr Projekt (also da, wo auch `build.gradle` liegt), und erstellen Sie dort ein Python-Skript `uebersetzungen-to-properties.py`, welches die CSV-Datei liest und für jede der enthaltenen Sprachen `xx` eine `message_xx.properties`-Dateien im von Spring dafür vorgesehenen Unterverzeichnis des Projekts generiert.

Sie können *nicht* davon ausgehen, dass die Reihenfolge der Sprach-Spalten oder deren Anzahl in der CSV-Datei bei Änderungen gleich bleibt, Sie dürfen also insbesondere *nicht* im Skript hardcoden, dass es nur drei Übersetzungen sind, diese in einer bestimmten Spalten stehen oder dass die Reihenfolge `de - en - nl` ist. Das Skript entnimmt die nötigen Informationen **ausschließlich** der ersten Datei-Zeile.

Mit Hilfe Ihres Skripts können Sie die Übersetzungen z.B. mit Excel oder LibreOffice Calc übersichtlich pflegen und bekommt stets aktuelle und konsistente `properties`-Dateien für beliebig viele Sprachen per Knopfdruck aus dem Skript. Die Unterstützung einer neuen Sprache erfordert dann nur das Einfügen einer weiteren Spalte in der CSV-Datei.

Denken Sie bitte auch daran, die Meldung bei Verstößen gegen Ihre `@GutesPassword`-Restriktion mehrsprachig anzubieten.

Benutzerprofil 17 bearbeiten

Name Größe muss zwischen 3 und 80 sein

E-Mail

Passwort

Geburtstag muss ein Datum in der Vergangenheit sein

☐ Ich mag (max. 5)
☐ Ich mag nicht (max. 5)

Edit user profile 17

Name size must be between 3 and 80

E-mail

Password

Date of birth must be a past date

☐ I like (max. 5)
☐ I don't like (max. 5)

Gebruikersprofiel 17 bewerken

Naam grootte moet tussen 3 en 80 liggen

E-mail

Wachtwoord

Geboortedatum moet te het verleden zijn

☐ Ik hou van (max. 5)
☐ Ik hou niet van (max. 5)

Sobald die Anwendung einmal durch den Umbau *internationalisiert* ist, kann sie ohne Codeänderung jederzeit nur durch Erzeugung weiterer `messages`-Dateien für neue Sprachen *lokalisiert* werden. Die Übersetzungen für die neue Sprache brauchen nur in der `uebersetzungen.csv`-Datei nachgetragen und das Generierungs-Skript damit gefüttert zu werden.

Bitte testen Sie auch, ob Ihre Fehlermeldungen in allen Sprachen angeboten werden. Bei Meldungen aus den Standard-Validatoren sollte dies automatisch der Fall sein. Für Ihre eigene `@GutesPassword`-Navigation können Sie das Standard-Attribut `messages` verwenden, um übersetzte Meldungen in gewohnter Form zu übergeben. Es ist *keine* Code-Änderung in Ihrem Validator erforderlich.

Aufgabe 4 (Kopf- und Fußzeile als Fragmente auslagern)

Perspektivisch werden wir mehr Ansichten als nur unser Benutzerformular haben, und da wäre eine einheitliche Kopf- bzw. Fußzeile schön, und noch schöner wäre es, wenn wir künftige Erweiterungen nur an einer Stelle machen müssten (Copy&Paste dieser Bereiche auf alle HTML-Seiten ist also keine gute Idee).

Thymeleaf bietet durch die Nutzung von Fragmenten eine Lösung. Bitte legen Sie innerhalb des `src/main/templates` einen Unterordner `fragments` an und legen Sie dort ein Thymeleaf-Template `kopffuss.html` an, welches eine Navigationsleiste mit Fragmentnamen `kopf` und eine Fußzeile mit Fragmentnamen `fuss` definiert. Bitte sehen Sie in der Kopfzeile einen Link mit dem Namen “Benutzer” sowie drei Sprach-Links für unsere Sprachen `DE`, `NL` und `EN` an (alle Links vorerst nur als Platzhalter ohne “Ziel”). In der Fußzeile können Sie einen “Impressum” und eine “Kontakt” Link vorsehen.

Bitte binden Sie Ihre Kopf- und Fußzeile nun in Ihr bestehendes `benutzerbearbeiten.html`-Template ein.

Benutzerprofil 17 bearbeiten

Name
Joghurta Biffel

E-Mail
joghurta@diebiffels.xy

Passwort
Passwort

Geburtstag
17.11.1991

Ich mag (max. 5)
Ich mag

Ich mag nicht (max. 5)
Ich mag nicht

ok

Impressum Kontakt

Aufgabe 5 (Language-Switch)

Ergänzen Sie bitte nun zum Abschluss noch die Möglichkeit, die Sprache direkt innerhalb der Anwendung zu wechseln, damit der Benutzer nicht immer an seinen Browsereinstellungen herumspielen muss.

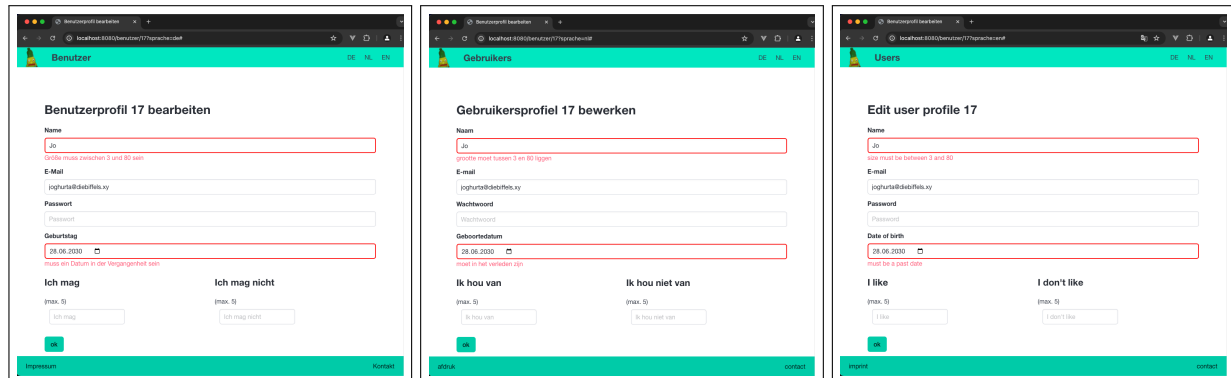
Passen Sie dazu bitte die Sprach-Umschalt-Links im Kopf-Fragment hinzu, um per Query-Parameter `sprache` entweder Deutsch (`de`), Englisch (`en`) oder Niederländisch (`nl`, und natürlich gerne weitere) auswählen zu können.

Benutzerprofil bearbeiten

localhost:8080/benutzer/17?sprache=de

Benutzer DE NL EN

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 Profilsansicht 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 in den Links **nicht** den URI-Pfad komplett hardcoden (und sollten dies auch nicht tun).

Hinweis: Da wie erwähnt bei der Sprachumstellung hier ein GET-Request auf die selbst Seite abgesetzt wird, sind eventuelle Validierungsmeldungen danach weg (die Formularinhalte sollten aber erhalten bleiben). Das ist ok. Wenn Sie das Formular erneut absenden, kommen die Validierungsmeldungen (hoffentlich in der neu ausgewählten Sprache) zurück.

Akzeptanzkriterien für diesen Projektstand:

- Falscheingaben bei Textfeldern werden nach dem Abschicken des Bearbeitungsformulars abgefangen und passende Fehlermeldungen beim jeweiligen Text-Eingabefeld ausgegeben.
 - Das Skript `uebersetzungen-zu-properties.py` erzeugt passende `messages`-Dateien für alle in `uebersetzungen.csv` hinterlegte Sprachen. Nach einem Neustart der Anwendung sind die Änderungen sichtbar. Eine Änderung der Übersetzungs-Spalten-Reihenfolge oder das Hinzufügen neuer Spalten für andere Sprachen beeinträchtigt die Funktionsweise des Skripts nicht. Die Default-Übersetzungsdatei `messages.properties` ist eine Kopie der für die erste Sprache (Spalte 2 in `uebersetzungen.csv`) erzeugten Datei.
 - Titel und Beschriftungen von Eingabefeldern im Benutzer-Formular werden standardmäßig gemäß Browser-Sprachpräferenz angezeigt.
 - Die Passwortheingabe wird mit dem eigenen Validator `@GutesPasswort` überprüft und fehlerhafte Eingaben werden mit Fehlermeldung abgewiesen.
 - Ein leeres Passwortfeld ergibt keinen Validierungsfehler.
 - Die Fehlermeldung für das Passwortfeld wird sprachabhängig gemäß der Übersetzung aus `uebersetzungen.csv` ausgegeben.
 - Durch Anhängen des Query-Parameters `sprache=...` wird die Benutzerformular-Seite in der gewünschten Sprache (`de`, `en`, `nl`) ausgegeben.
 - Durch die Sprachauswahl-Links in der Kopfzeile des Benutzerformulars kann deren Anzeigesprache unabhängig von der Browser-Präferenz umgestellt werden.
-