Spring MVC. Validacija

Hibernate Validator instaliavimas

Parsisiųskime ir instaliuokime Hibernate validator:

```
<dependency>
   <groupId>org.hibernate.validator</groupId>
   <artifactId>hibernate-validator</artifactId>
   <version>6.0.13.Final</version>
</dependency>
```

Laukų tikrinimo procesas

Kuriant laukų validaciją turėsime atlikti:

1. Sukurti reikalavimus ir tikrinimo taisykles Entities klasėse
2. Sukurti klaidos pranešimų atvaizdavimą HTML formose
3. Įdėjimo metu atlikti validaciją
4. Atnaujinti kontrolerio metodą kuris saugo informaciją

1. Reikalavimai ir tikrinimo taisykles Entities klasėse

Reikalavimai laukams nurodomi entities klasėse virš laukelių anotacijų pavidalu.

Anotacija	Aprašas
@NotNull	Reikšmė ne null
@Min	Skaičius turi būti didesnis, arba lygus
@Max	Skaičius turi būti mažesnis, arba lygus
@Size	Dydis privalo būti toks koks nurodytas
@Pattern	Pagal reg. Expression
@Future / @Past	Datos tikrinimas
Kitos	

Užduotis

Padarykime kad laukas name (Task klasėje) būtų privalomas ir ne trumpesnis nei 6 simboliai.

Tai galėtume atlikti Task Entity klasėje virš lauko name įdėkime tokią anotaciją:

```
@NotNull(message="laukas yra privalomas")
@Size(min=6, message="pavadinimas ne trumpesinis nei 6 simboliai ")
```

2. klaidos pranešimo atvaizdavimas

Įvykus klaidai, norint formoje atvaizduoti pranešimus, mes galime pasinaudoti spring formų biblioteka:

```
<form:errors path="laukelio_pavadinimas" />
```

Vykdymo metu , įvykus klaidai, tai sugeneruos div žymą su viduje patalpintu klaidos aprašu.

Užduotis

Padarykime jog įvedimo metu nejvedus užduoties pavadinimo, arba įvedus pavadinimą per trumpą, sistema parodytų klaidą (po įvedimo lauku).

Klaidai panaudokime standartinį bootstrap elementą:

```
<div class=" alert alert-danger">Klaidos pranešimas</div>
```

3. Validatoriaus vykdymas

Norėdami jog prieš atliekant išsaugojimo veiksmą būtų iškviestas validatorius, prieš Entity kurį tikrinsime turime nurodyti anotaciją: **`@Valid`**

Pavyzdžiui:

```
public String storeTask(@Valid @ModelAttribute("task") Task task) {...}
```

Tikrinimo atsakymai bus įkelti į BindingResult klasės objektą, kurį privalome nurodyti iškarto po modelio atributo kurį mes tikrinsime:

```
public String storeTask(@Valid @ModelAttribute("task") Task task,
                        BindingResult bindingResult ) { }
```

Metodas `bindingResult.hasErrors()` gražintų true, jei bus neįgyvendinta bent viena tikrinimo taisyklė

Užduotis

Padarykime taip, jog pridedant naują įrašą patikrintų ar jis tenkina reikalavimus ir tuomet arba atvaizduotų atgal tą pačią formą, arba įkeltu įrašą ir parodytų užduočių sąrašą

Tai galėtume pasiekti panaudodami sąliginį sakinį:

```
if (bindingResult.hasErrors()) {
   return "addTask";          //Atvaizduosime tą patį sąrašą
}else {
   //Pridėsime naują įrašą
   return "redirect:/list";  //Pereisime į sąrašą
}
```

Tarpai įvedimo laukuose

Tarpai įvedimo laukuose užskaitomi kaip įprasti simboliai, todėl pavadinimas vien tik iš tarpų, pas mus taip pat būtų validus.

Norėdami tai ištaisyti galime pasinaudoti Spting InitBinder anotacija (ji prijungia kontrolerio funkcijas ir jas įvykdo prieš paduodant duomenis į kontrolerio metodą (pagal route)).

Todėl, kontrolerį galėtume papildyti metodu:

```
@InitBinder
public void initBinder(WebDataBinder dataBinder){
  StringTrimmerEditor stEditor = new StringTrimmerEditor(true);
  dataBinder.registerCustomEditor(String.class, stEditor);
}
```

Užduotis

Papildykime mūsų projekto kontrolerį metodu, kuris užkrovimo metu įjungs String Trimm klasę kuri:

- 1. pašalins tarpo simbolius iš įvedimo lauko priekio ir pabaigos
- 2. tuščius laukus (kuriuos sudaro vien tik tarpai) pakeis į null

Užduotis

Pamėginkite atnaujinti užduoties įrašą ir laukelyje pavadinimas neįrašykite nieko. Programa turėtų nulūžti, kadangi mes nurodėme taisyklę, kad laukas yra privalomas, tačiau neatlikome tikrinimo ir mėginome įterpti tuščią lauką.

Pamėginkite pridėti šio lauko tikrinimą ir atliekant atnaujinimą, taip pat pakoreguokite atnaujinimo formą taip, kad ji rodytų pranešimą apie klaidą.

Užduotis

Mūsų užduotys turės tam tikrus prioritetus (skaičius nuo 1 iki 10):

1 – užduotis kurią reikia atlikti tuoj pat

10 – užduotis kuri gali palaukti

1. Papildykime mūsų duomenų bazę įrašu priority - integer
2. Pridėkime šį įrašą mūsų entity klasėje
3. pridėkime šio įrašo įvedimą mūsų įvedimo formoje
4. atvaizduokime šį įrašą mūsų redagavimo formoje

Užduotis

Apribokime kad šio įrašo reikšmė būtų ne mažesnė už 1 ir ne didesnė už 10.

Tai galime padaryti su validacijos taisyklėmis:

@Min(value=1, message="pranešimas")

@Max(value=1, message="pranešimas")

Atlikime tikrinimą, padarykime formose klaidos pranešimo atvaizdavimą

Mūsų pranešimai

Jei pamėgintumėme į skaičiaus laiką įvesti tekstą, gautumėme pranešimą apie netinkamą skaičių.

Pamėginkime pakeisti pranešimo tekstą į mūsų tekstą.

Tam sukurkime katalogą ir failą:

/src/resources/messages.properties

Faile messages.properties patalpinkime eilutę aprašančią mūsų pranešimą:

typeMismatch.task.priority=Privalo būti skaičius nuo 1 iki 10

Pranešimų failo užkrovimas

Norėdami jog Spring užkrautų mūsų sukurtą failą, Spring servleto konfigūraciniame xml faile, turime nurodyti:

```
<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">

  <property name="basenames" value="resources/messages">

</bean>
```