

FAQ: How to make Integer field required and handle Strings: freePasses

FAQ: How to Make Integer field required and handle Strings: freePasses?

Question:

I am getting the following error when i submit the form with an empty value for customer "freePasses". I am using @NotNull on the field "freePasses". It is not throwing "is required" after validation after submit.

How to fix this? Please help.

Also, how do I handle validation if the user enters String input for the integer field?

Answer:

Great question!

The root cause is the freePasses field is using a primitive type: int. In order to check for null we must use the appropriate wrapper class: **Integer**.

To resolve this, In Customer.java, replace "int" with "Integer"

```
@NotNull(message="is required")
@Min(value=0, message="must be greater than or equal to zero")
@Max(value=10, message="must be less than or equal to 10")
private Integer freePasses;
```

Then update your getter/setter methods to use "Integer"

```
public Integer getFreePasses() {  
    return freePasses;  
}  
  
public void setFreePasses(Integer freePasses) {  
    this.freePasses = freePasses;  
}
```

Save everything and retest.

=====

Here is the full source code.

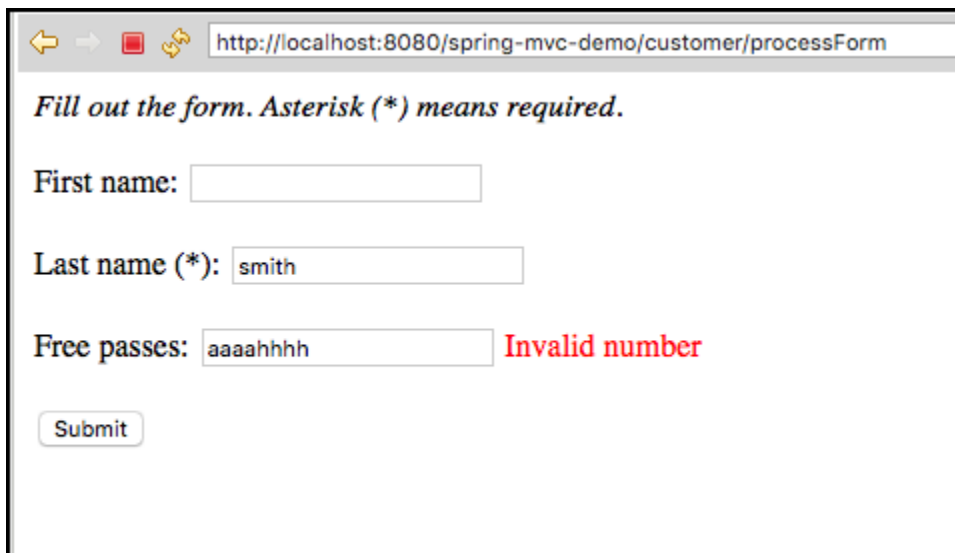
```
1. package com.luv2code.springdemo.mvc;  
2. import javax.validation.constraints.Max;  
3. import javax.validation.constraints.Min;  
4. import javax.validation.constraints.NotNull;  
5. import javax.validation.constraints.Pattern;  
6. import javax.validation.constraints.Size;  
7.  
8. public class Customer {  
9.  
10.     private String firstName;  
11.  
12.     @NotNull(message="is required")  
13.     @Size(min=1, message="is required")  
14.     private String lastName;  
15.  
16.     @NotNull(message="is required")  
17.     @Min(value=0, message="must be greater than or equal to zero")  
18.     @Max(value=10, message="must be less than or equal to 10")  
19.     private Integer freePasses;  
20.  
21.     @Pattern(regexp="^[a-zA-Z0-9]{5}", message="only 5 chars/digits")  
22.     private String postalCode;  
23.  
24.  
25.     public String getPostalCode() {  
26.         return postalCode;  
27.     }  
28.  
29.     public void setPostalCode(String postalCode) {  
30.         this.postalCode = postalCode;  
31.     }  
32.  
33.     public String getFirstName() {  
34.         return firstName;  
35.     }  
36.  
37.     public void setFirstName(String firstName) {  
38.         this.firstName = firstName;
```

```
39.     }
40.
41.     public String getLastName() {
42.         return lastName;
43.     }
44.
45.     public void setLastName(String lastName) {
46.         this.lastName = lastName;
47.     }
48.
49.     public Integer getFreePasses() {
50.         return freePasses;
51.     }
52.
53.     public void setFreePasses(Integer freePasses) {
54.         this.freePasses = freePasses;
55.     }
56.
57. }
```

====

Handle String Input for Integer Fields

If the user enters String input such as "abcde" for the Free Passes integer field, we'd like to give a descriptive error message.



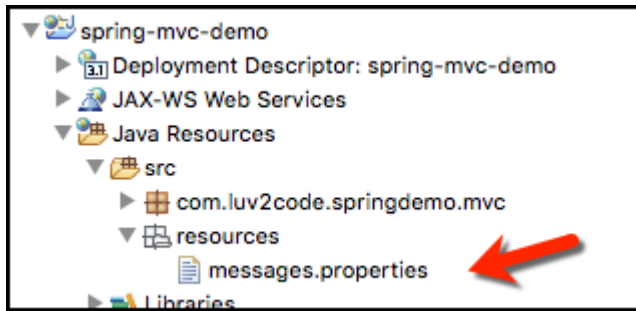
A screenshot of a web browser window showing a form titled "Fill out the form. Asterisk (*) means required." The form has three input fields: "First name:" (empty), "Last name (*):" (containing "smith"), and "Free passes:" (containing "aaaahhhh"). To the right of the "Free passes:" field, there is a red error message that says "Invalid number". Below the fields is a "Submit" button. The browser's address bar shows the URL "http://localhost:8080/spring-mvc-demo/customer/processForm".

We basically need to override the default Spring MVC validation messages.

Follow these steps.

1. In your Eclipse project, expand the node: **Java Resources**
2. Right-click the **src** directory and create a new sub-directory: **resources**
3. Right-click the **resources** sub-directory and create a new file named: **messages.properties**

Your directory structure should look like this:



4. Add the following entry to the **messages.properties** file

```
typeMismatch.customer.freePasses=Invalid number
```

5. Save file

This file contains key/value pairs for the error message type

For a basic example:

```
typeMismatch.customer.freePasses=Invalid number
```

The format of the error key is: *code* + "." + *object name* + "." + *field*

To find out the given error code, in your Spring controller, you can log the details of the binding result

```
System.out.println("Binding result: " + theBindingResult);
```

For details, see the docs [here](#)

- <http://docs.spring.io/spring/docs/current/javadoc-api/org.springframework.validation.DefaultMessageCodesResolver.html>

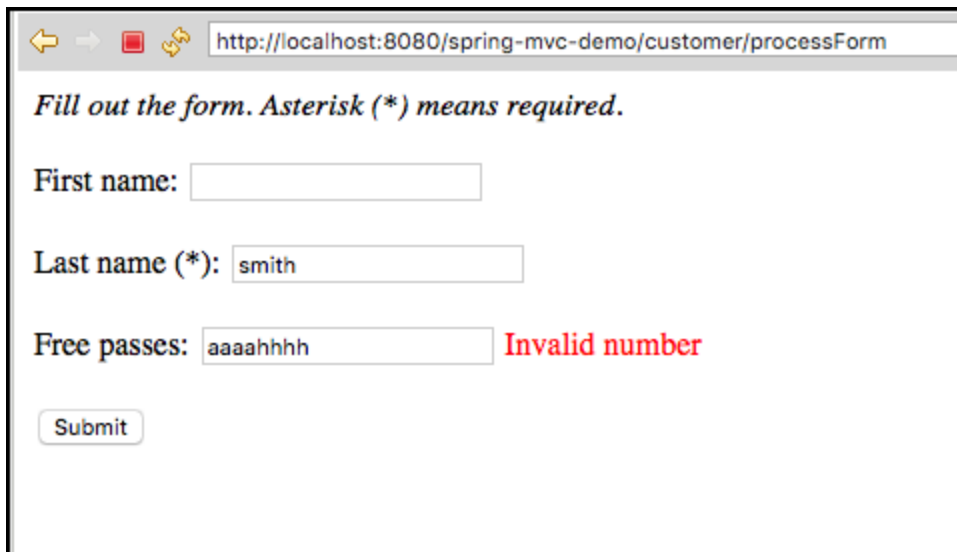
6. Edit your config file: **WEB-INF/spring-mvc-demo-servlet.xml**

Add the following:

```
1. <bean id="messageSource"
2.     class="org.springframework.context.support.ResourceBundleMessageSource">
3.
4.     <property name="basenames" value="resources/messages" />
5.
6. </bean>
```

7. Save the file. Restart the Tomcat server

8. Run your app and add bad data for the "Free Passes" field. You will see the error message from our properties file.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/spring-mvc-demo/customer/processForm`. The page content includes a heading *Fill out the form. Asterisk (*) means required.* and three form fields. The first field is labeled "First name:" and is empty. The second field is labeled "Last name (*):" and contains the text "smith". The third field is labeled "Free passes:" and contains the text "aaaahhhh". To the right of this field, the text "Invalid number" is displayed in red. Below the fields is a "Submit" button.

Resources for this lecture

- [solution-code-spring-mvc-validation-handle-strings-for-integer-fields.zip](#)