مثالی برای Error Handling برای Spring MVC

در این مثال چگونگی استفاده از Error Handling و Validator ها در فرم ها با استفاده از Spring Web MVC Framework نشان داده شده است. برای شروع Eclipse IDE را آماده کنید و جهت توسعه ی یک برنامه ی وب پویا با استفاده از Spring Web Framework مراحل زیر را دنبال کنید.

مرحله	توضیحات
1	پروژه ای با نام HelloWeb در بسته ی com.tutorialspoint همان طور که در بخش Spring MVC - Hello World توضیح داده شده است، ایجاد کنید.
2	در بسته ی com.tutorialspoint کلاس های جاوای Student ، StudentValidator و StudentController را ایجاد کنید.
3	در زیر پوشه ی jsp فایل های ویوی result.jsp و addStudent.jsp را ایجاد کنید.
4	مرحله ی نهایی ایجاد محتوای فایل های منبع و پیکربندی و اکسپورت کردن برنامه به صورت زیر است.

Student.java

```
package com.tutorialspoint;


public class Student {
  private Integer age;
  private String name;
  private Integer id;

  public void setAge(Integer age) {
    this.age = age;
  }
```

```
  public Integer getAge() {
    return age;
  }

  public void setName(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }

  public void setId(Integer id) {
    this.id = id;
  }
  public Integer getId() {
    return id;
  }
}
```

StudentValidator.java

```
package com.tutorialspoint;

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

public class StudentValidator implements Validator {
```

```
  @Override
  public boolean supports(Class<?> clazz) {
    return Student.class.isAssignableFrom(clazz);
  }

  @Override
  public void validate(Object target, Errors errors) {
    ValidationUtils.rejectIfEmptyOrWhitespace(errors,
      "name", "required.name","Field name is required.");
  }
}
```

StudentController.java

```
package com.tutorialspoint;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Validator;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;


@Controller

public class StudentController {


  @Autowired

  @Qualifier("studentValidator")

  private Validator validator;


  @InitBinder

  private void initBinder(WebDataBinder binder) {

    binder.setValidator(validator);

  }


  @RequestMapping(value = "/addStudent", method = RequestMethod.GET)

  public ModelAndView student() {

    return new ModelAndView("addStudent", "command", new Student());

  }


  @ModelAttribute("student")

  public Student createStudentModel() {

    return new Student();

  }


  @RequestMapping(value = "/addStudent", method = RequestMethod.POST)

  public String addStudent(@ModelAttribute("student") @Validated Student student,
```

```
    BindingResult bindingResult, Model model) {


    if (bindingResult.hasErrors()) {
      return "addStudent";
    }
    model.addAttribute("name", student.getName());
    model.addAttribute("age", student.getAge());
    model.addAttribute("id", student.getId());


    return "result";
  }
}
```

HelloWeb-servlet.xml

```
<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:context = "http://www.springframework.org/schema/context"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd">


  <context:component-scan base-package = "com.tutorialspoint" />


  <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name = "prefix" value = "/WEB-INF/jsp/" />
```

```
   <property name = "suffix" value = ".jsp" />
  </bean>

  <bean id = "studentValidator" class = "com.tutorialspoint.StudentValidator" />
</beans>
```

در اینجا در اولین متد سرویس student() شیء Student خالی را در شیء ModelAndView با نام "command" عبور داده ایم. دلیل انجام این کار این است که spring framework انتظار دارد اسم شیء "command"باشد. این برای حالتی است که ما از تگ های<form:form> در فایل JSP استفاده کنیم. بنابراین زمانی که متد student() فراخوانی می شود، این متد ویوی addStudent.jsp را برگشت می دهد.

دومین متد سرویس addStudent() در برابر یک متد POST و در آدرس HelloWeb/addStudent فراخوانی می شود. شما باید شیء مدل خود را بر اساس اطلاعات ارائه شده آماده کنید. در نهایت یک ویوی "result" از متد سرویس برگشت داده می شود. این امر باعث می شود result.jsp نمایش داده شود. در صورتی که در حین استفاده از validator با خطا مواجه شدید، آن گاه view یکسان "addStudent" برگشت داده می شود و Spring به صورت خودکار پیام های خطا را از BindingResult در view تزریق می کند.

addStudent.jsp

```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
<html>
  <head>
    <title>Spring MVC Form Handling</title>
  </head>
  <style>
    .error {
      color: #ff0000;
    }
```

```
  .errorblock {
    color: #000;
    background-color: #ffEEEE;
    border: 3px solid #ff0000;
    padding: 8px;
    margin: 16px;
  }
</style>

<body>
  <h2>Student Information</h2>
  <form:form method = "POST" action = "/HelloWeb/addStudent" commandName = "student">
  <form:errors path = "*" cssClass = "errorblock" element = "div" />
    <table>
      <tr>
        <td><form:label path = "name">Name</form:label></td>
        <td><form:input path = "name" /></td>
        <td><form:errors path = "name" cssClass = "error" /></td>
      </tr>
      <tr>
        <td><form:label path = "age">Age</form:label></td>
        <td><form:input path = "age" /></td>
      </tr>
      <tr>
        <td><form:label path = "id">id</form:label></td>
        <td><form:input path = "id" /></td>
```

```
      </tr>
      <tr>
        <td colspan = "2">
        <input type = "submit" value = "Submit"/>
        </td>
      </tr>
    </table>
  </form:form>
 </body>
</html>
```

در اینجا ما برای نمایش پیام های خطا از تگ < form:errors /> به همراه path="*" استفاده کرده ایم. برای مثال:

```
<form:errors path = "*" cssClass = "errorblock" element = "div" />
```

این کار باعث می شود برای ارزیابی تمامی ورودی ها، پیام های خطا نمایش داده شوند.

در اینجا ما برای نمایش پیام های خطا از تگ < form:errors /> به همراه path="*" استفاده کرده ایم. برای مثال:

```
<form:errors path = "name" cssClass = "error" />
```

این کار باعث می شود پیام های خطا برای ارزیابی فیلد اسم نمایش داده شوند.

result.jsp

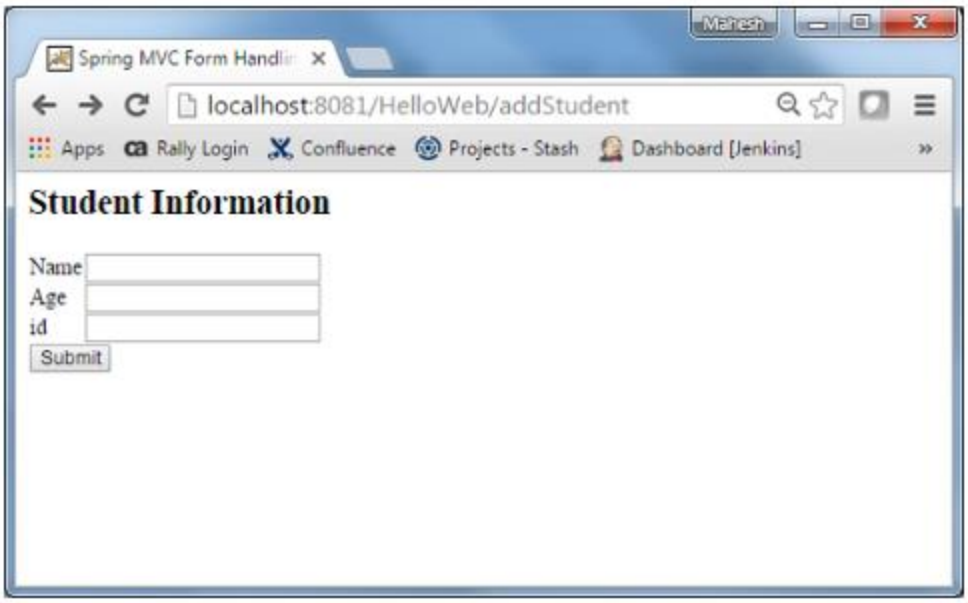
```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
<html>
 <head>
  <title>Spring MVC Form Handling</title>
 </head>
```

```
 <body>


  <h2>Submitted Student Information</h2>
  <table>
    <tr>
      <td>Name</td>
      <td>${name}</td>
    </tr>
    <tr>
      <td>Age</td>
      <td>${age}</td>
    </tr>
    <tr>
     <td>ID</td>
     <td>${id}</td>
    </tr>
  </table>
 </body>
</html>
```

بعد از آن که کار فایل های پیکربندی و منبع تمام شد، برنامه ی خود را اکسپورت کنید. بر روی برنامه ی خود کلیک راست کنید، از گزینه ی Export → WAR File استفاده کنید و فایل HelloWeb.war را داخل پوشه ی webapps متعلق به Tomcat ذخیره کنید.

حالا سرور Tomcat را اجرا کنید و مطمئن شوید که از طریق پوشه ی webapps و با استفاده از یک مرورگر استاندارد می توانید به دیگر صفحات وب دسترسی پیدا کنید. حالا در صورت وارد کردن آدرس http://localhost:8080/ HelloWeb/addstudent و در صورت نبود مشکل در برنامه ی Spring Web صفحه ی زیر نمایش داده می شود.



بعد از وارد کردن اطلاعات مورد نیاز، بر روی دکمه ی submit کلیک کنید تا اگر مشکلی در برنامه ی Spring Web وجود نداشته باشد، صفحه زیر نمایش داده شود.