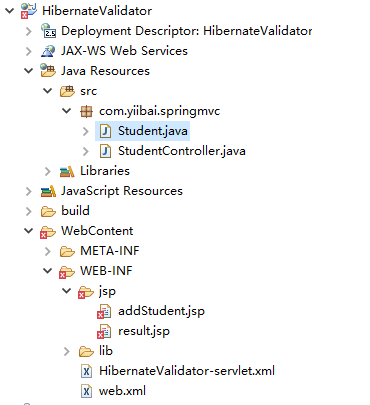
下面的示例演示如何使用**Spring Web MVC**框架在表单中使用错误处理和验证器。 首先使用Eclipse IDE，并按照以下步骤使用**Spring Web Framework**开发基于动态表单的Web应用程序：

1. 创建一个名称为 **HibernateValidator** 的动态WEB项目。
2. 在 com.yiibai.springmvc 包下创建三个Java类：Student和StudentController 。
3. 在jsp子文件夹下创建两个视图文件：addStudent.jsp, result.jsp。
4. 下载Hibernate Validator库 - [Hibernate Validator](http://hibernate.org/validator/)。解压缩hibernate-validator-5.3.4.Final.jar和所需的依赖关系存并将它们放在CLASSPATH中。
5. 在src文件夹下创建属性文件messages.properties。
6. 最后一步是创建所有源和配置文件的内容并运行应用程序，详细如下所述。

完整的项目文件目录结构如下所示 -



**Student.java** 的代码如下所示 -

package com.yiibai.springmvc;

import org.hibernate.validator.constraints.NotEmpty;

import org.hibernate.validator.constraints.Range;

public class Student {

@Range(min = 1, max = 150)

private Integer age;

@NotEmpty

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;

}

}

Java

**StudentController.java** 的代码如下所示 -

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

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 {

@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";

}

}

Java

**message.properties** 配置如下所示 -

NotEmpty.student.name = Name is required!

Range.student.age = Age value must be between 1 and 150!

这里的键可以是<Annotation>.<object-name>.<attribute>。Value是要显示的消息。

**HibernateValidator-servlet.xml** 配置如下所示 -

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

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

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

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

<mvc:annotation-driven />

<bean class="org.springframework.context.support.ResourceBundleMessageSource"

id="messageSource">

<property name="basename" value="messages" />

</bean>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/" />

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

</bean>

</beans>

XML

这里的第一个服务方法student()，已经在ModelAndView对象中传递了一个名称为“command”的空对象，因为如果在JSP文件中使用<form：form>标签，**spring**框架需要一个名称为“command”的对象。 所以当调用student()方法时，返回addStudent.jsp视图。

第二个服务方法addStudent()将在URL： HibernateValidator/addStudent 上的POST方法被调用。将根据提交的信息准备模型对象。 最后从服务方法返回“result”视图，这将渲染result.jsp。 如果使用validator生成错误，则返回相同的视图“addStudent”，则Spring自动从视图中的BindingResult注入错误消息并显示出来。

**addStudent.jsp** 的代码如下所示 -

<%@ page contentType="text/html; charset=UTF-8" %>

<%@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>学生信息</h2>

<form:form method="POST" action="/HibernateValidator/addStudent" commandName="student">

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

<table>

<tr>

<td><form:label path="name">姓名：</form:label></td>

<td><form:input path="name" /></td>

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

</tr>

<tr>

<td><form:label path="age">年龄：</form:label></td>

<td><form:input path="age" /></td>

<td><form:errors path="age" cssClass="error" /></td>

</tr>

<tr>

<td><form:label path="id">编号：</form:label></td>

<td><form:input path="id" /></td>

</tr>

<tr>

<td colspan="2">

<input type="submit" value="提交"/>

</td>

</tr>

</table>

</form:form>

</body>

</html>

HTML

上面的代码中使用了<form：errors />标记，其中path =“\*”来呈现错误消息。例如-

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

Java

它将呈现所有输入验证的错误消息。  
使用带有path =“name”的<form：errors />标记来渲染name字段的错误消息。例如 -

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

<form:errors path="age" cssClass="error" />

Java

它将呈现姓名(name)和年龄(age)字段验证的错误消息。

**result.jsp** 的代码如下所示 -

<%@ page contentType="text/html; charset=UTF-8" %>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>Spring MVC Form Handling</title>

</head>

<body>

<h2>提交的学生信息如下 -</h2>

<table>

<tr>

<td>姓名：</td>

<td>${name}</td>

</tr>

<tr>

<td>年龄：</td>

<td>${age}</td>

</tr>

<tr>

<td>编号：</td>

<td>${id}</td>

</tr>

</table>

</body>

</html>

HTML

完成创建源和配置文件后，发布应用程序到Tomcat服务器。

现在启动Tomcat服务器，当访问URL => [http://localhost:8080/HibernateValidator/addStudent](http://localhost:8080/HibernateValidator/addStudent" \t "_blank) ， 如果**Spring Web**应用程序没有问题，应该看到以下结果：

