Additional Spring MVC Topics

- Spring MVC JSP Tags
- Validation
- Redirection

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Spring MVC JSP Tags

- Spring MVC provides a JSP **tag library** that you can use in view pages
- The tags are especially useful in pages with HTML forms

```
Tag         Description
---         -----------
form        Defines a form that participates in Spring MVC
input       Generates an <input type="text"> widget
password    Generates an <input type="password> widget
checkbox    Genaterates an <input type="checkbox"> widget
. . .
```


Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Defining a Form with Spring MVC Tags

- You can use **modelAttribute** to reference a bean that your controller manipulates and then use **path** to reference its properties

```
1    <%@ taglib prefix="form"
2      uri="http://www.springframework.org/tags/form" %>
3    . . .
4    <form:form modelAttribute="student" method="post">
5      Name: <form:input path="name"/><p>
6      ID:   <form:input path="studentID"/><p>
7      GPA:  <form:input path="gpa"/><p>
8      <input type="submit" value="Register">
9    </form:form>
```

JSP tag libraries are a way to extend the syntax available to JSPs. To use the custom tags, you must write a "taglib" directive that specifies a prefix and the name (URL) of the library. You can use any prefix you want.

The "path" value must match properties defined in the bean as get/set methods.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Controller with Model Attribute

```
1    @Controller
2    public class RegisterStudentController {
3      @RequestMapping(value="/register",
4         method = RequestMethod.GET)
5      public String setupForm(Model model) {
6        Student s = new Student();
7        s.setStudentID(1);
8        model.addAttribute("student", s);
9        return "studentForm";
10     }
11
12     @RequestMapping(value="/register",
13        method = RequestMethod.POST)
14     public String submitForm(@RequestParam("student")
15         Student student) {
16       . . .
17     }
18   }
```

We stored a student bean, with properties of "studentID", "name" and "gpa" as a model attribute. Note that the name "student" and the property names match the JSP shown on the previous page.

Note that in the submitForm() method, we don't really need the @RequestParam annotation since we assigned the parameter the name "student", which matches the name of the model attribute.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is the Bean Validation Framework?

- **Bean Validation** (JSR 303) is a Java standard for defining validation on JavaBeans
- Bean Validation is a standard part of JEE Version 6 and later and is supported by Spring
- Bean Validation provides an annotation-based, standard approach to centralize validation rules

JSF 2.x and later implements also support Bean Validation.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Validation Example: The Bean

```
1    public class Student
2    {
3       @NotNull(message="ID can't be empty")
4       private int studentID;
5
6       @DecimalMin(value="0.0", message="Too small!")
7       @DecimalMax(value="4.0", message="Too big!")
8       private double gpa;
9
10      @Size(min=1, max=300, message="Name length bad!")
11      private String name;
12
13      . . .
14   }
```

You can put the annotations either on fields as shown here, or on a "get" method.

These annotations are from the javax.validation package.

Note that you are not required to supply an error message; Spring MVC provides reasonable defaults if you don't. You can also externalize the messages to a "properties" file so you can avoid hard-coding them in code and to make it easier to support other languages, such as German or Spanish.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Bean Validation Annotations

Annotation	Description	Example
@AssertTrue	The boolean must be true	@AssertTrue private boolean f;
@Future	The date must be in the future	@Future private Date tomorrow;
@Min	The integer's minimum value	@Min(1) private int quantity;
@Pattern	The string must match a regular expression	@Pattern(regexp="4[0-9]{15}") private String credCard;

This table does not show all of the constraint annotations. For a more complete listing, see:

http://download.oracle.com/javaee/6/tutorial/doc/gircz.html

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Using Bean Validation with Spring MVC

- Configure your Spring bean(s) with validation annotations
- Use <form:errors> tags in your JSP
- Use **@Valid** in your controller's "submit" method
- If errors, controller should send user back to original page

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JSP With Validation Tags

```
1    <%@ taglib prefix="form"
2      uri="http://www.springframework.org/tags/form" %>
3    . . .
4    <form:form modelAttribute="student" method="post">
5      <p>
6        Name: <form:input path="name"/>
7        <form:errors path="name"/>
8      <p>
9     . . .
10   </form:form>
```


The <form:errors> tag only displays a message if the controller "queues" messages and sends user back to the JSP. In this case, we displayed error messages right alongside the offending input widget, but you can also display a summary list of error messages using a path of "*".

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Controller with Validation

```
1    @Controller
2    public class RegisterStudentController {
3
4    . . .
5
6      @RequestMapping(value="/register",
7         method = RequestMethod.POST)
8      public String submitForm(@RequestParam("student")
9        @Valid Student student, BindingResult result) {
10
11      if(!result.hasErrors()) {
12         . . .
13        return "registerSuccess";
14      }
15      else
16       return "studentForm";
17     }
18   }
```

The @Valid annotation on the model-attribute object causes Spring to apply the validation rules defined in the Student bean class to the data entered by the user into the form. If any validations fail, Spring queues up an error message in the BindingResult, which we can check before returning a navigation destination.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Post/Redirect/Get

- After the successul posting a form, it's common to redirect to another page - this pattern is referred to as **Post/Redirect/Get**
- Spring MVC supports redirection using the **redirect:** prefix on a navigation destination

```
1    if(!result.hasErrors()
2    {
3      // do business logic
4
5      return "redirect:/registerSuccess";
6    }
7    else
8    {
9       return "studentForm";
10   }
```


Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Complete Validation Example

Student.java

studentForm.jsp

RegisterStudentController.java

registerSuccess.jsp

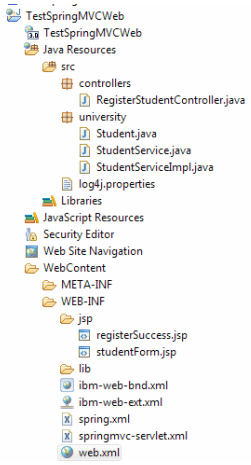
StudentService.java

StudentServiceImpl.java

spring.xml

springmvc-servlet.xml

web.xml



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
1    package university;
2
3    import java.io.Serializable;
4
5    import javax.validation.constraints.DecimalMax;
6    import javax.validation.constraints.DecimalMin;
7    import javax.validation.constraints.Min;
8    import javax.validation.constraints.NotNull;
9    import javax.validation.constraints.Size;
10
11   @SuppressWarnings("serial")
12   public class Student implements Serializable
13   {
14
15       @NotNull(message="ID must not be empty")
16       @Min(value=1, message="ID must be greater than zero")
17       private int studentID;
18
19       @NotNull(message="Name must not be empty")
20       @Size(min=2, max=30,
21         message="Name must be at least two characters, but no more than 30")
22       private String name;
23
24       @DecimalMin(value="0.0",
25         message="GPA must be greater than or equal 0.0")
26       @DecimalMax(value="4.0",
27         message="GPA must be less than or equal 4.0")
28       private double gpa;
29
30       public int getStudentID()
31       {
32           return studentID;
33       }
34
35       public void setStudentID(int studentID)
36       {
37           this.studentID = studentID;
38       }
39
40       public String getName()
41       {
42           return name;
43       }
44
45       public void setName(String name)
```

```
46      {
47          this.name = name;
48      }
49
50      public double getGpa()
51      {
52          return gpa;
53      }
54
55      public void setGpa(double gpa)
56      {
57          this.gpa = gpa;
58      }
59
60      @Override
61      public String toString()
62      {
63          return "Student [studentID=" + studentID + ", name=" + name + ", gpa="
64                  + gpa + "]";
65      }
66
67  }
```

```
1    <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
2
3    <!DOCTYPE html>
4    <html>
5      <head>
6        <title>Register a Student</title>
7        <style>
8          table {border: 0px;}
9          .error {color: red;}
10       </style>
11     </head>
12     <body>
13       <h1>Register a Student</h1>
14       <form:form modelAttribute="student" method="post">
15         <table>
16           <tr>
17             <td>Name: </td>
18             <td>
19               <form:input path="name"/>
20               <form:errors path="name" cssClass="error"/>
21             </td>
22           </tr>
23           <tr>
24             <td>ID: </td>
25             <td>
26               <form:input path="studentID"/>
27               <form:errors path="studentID" cssClass="error"/>
28             </td>
29           </tr>
30           <tr>
31             <td>GPA: </td>
32             <td>
33               <form:input path="gpa"/>
34               <form:errors path="gpa" cssClass="error"/>
35             </td>
36           </tr>
37         </table>
38         <input type="submit" value="Register">
39       </form:form>
40     </body>
41   </html>
```

```
1    package controllers;
2
3    import javax.validation.Valid;
4
5    import org.springframework.beans.factory.annotation.Autowired;
6    import org.springframework.stereotype.Controller;
7    import org.springframework.ui.Model;
8    import org.springframework.validation.BindingResult;
9    import org.springframework.web.bind.annotation.RequestMapping;
10   import org.springframework.web.bind.annotation.RequestMethod;
11
12   import university.Student;
13   import university.StudentService;
14
15   @Controller
16   public class RegisterStudentController
17   {
18       @Autowired
19       private StudentService service;
20
21       @RequestMapping(value="/register", method = RequestMethod.GET)
22       public String setupForm(Model model)
23       {
24           Student s = new Student();
25           s.setGpa(4.00);
26           s.setName("No name");
27           s.setStudentID(1);
28           model.addAttribute("student", s);
29           return "studentForm";
30       }
31
32       @RequestMapping(value="/register", method = RequestMethod.POST)
33       public String submitForm(@Valid Student student,
34               BindingResult result)
35       {
36           System.out.println("In submitForm student: " + student);
37           if(!result.hasErrors())
38           {
39               service.registerStudent(student);
40
41               // We include the "redirect:" prefix so the controller
42               // redirects instead of forwarding
43               return "redirect:/registerSuccess";
44           }
45           else
```

```
46          {
47              System.out.println("There were validation errors!" + result);
48              return "studentForm";
49          }
50      }
51
52      @RequestMapping(value="/registerSuccess", method = RequestMethod.GET)
53      public String registerSuccess()
54      {
55          return "registerSuccess";
56      }
57  }
```

```
1    <!DOCTYPE html>
2    <html>
3      <head>
4        <title>Success</title>
5      </head>
6      <body>
7        <h1>Student Registered Successfully</h1>
8      </body>
9    </html>
```

```
1    package university;
2
3    public interface StudentService
4    {
5        public void registerStudent(Student s);
6    }
```

```java
package university;

import org.springframework.stereotype.Component;

@Component
public class StudentServiceImpl implements StudentService
{
    @Override
    public void registerStudent(Student s)
    {
        System.out.println("Registering the student: " + s);
    }
}
```

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <beans xmlns="http://www.springframework.org/schema/beans"
3        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4        xmlns:aop="http://www.springframework.org/schema/aop"
5        xmlns:context="http://www.springframework.org/schema/context"
6        xmlns:jee="http://www.springframework.org/schema/jee"
7        xmlns:jms="http://www.springframework.org/schema/jms"
8        xmlns:lang="http://www.springframework.org/schema/lang"
9        xmlns:tx="http://www.springframework.org/schema/tx"
10       xmlns:util="http://www.springframework.org/schema/util"
11       xsi:schemaLocation="http://www.springframework.org/schema/aop
12           http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
13           http://www.springframework.org/schema/beans
14           http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
15           http://www.springframework.org/schema/context
16           http://www.springframework.org/schema/context/spring-context-3.1.xsd
17           http://www.springframework.org/schema/jee
18           http://www.springframework.org/schema/jee/spring-jee-3.1.xsd
19           http://www.springframework.org/schema/lang
20           http://www.springframework.org/schema/lang/spring-lang-3.1.xsd
21           http://www.springframework.org/schema/tx
22           http://www.springframework.org/schema/tx/spring-tx-3.1.xsd
23           http://www.springframework.org/schema/util
24           http://www.springframework.org/schema/util/spring-util-3.1.xsd">
25
26     <context:component-scan base-package="university"/>
27
28
29   </beans>
30
```

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <beans xmlns="http://www.springframework.org/schema/beans"
3        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.sp
4        xmlns:context="http://www.springframework.org/schema/context"
5        xmlns:jee="http://www.springframework.org/schema/jee" xmlns:jms="http://www.sp
6        xmlns:lang="http://www.springframework.org/schema/lang" xmlns:tx="http://www.s
7        xmlns:util="http://www.springframework.org/schema/util"
8        xmlns:mvc="http://www.springframework.org/schema/mvc"
9        xsi:schemaLocation="http://www.springframework.org/schema/aop
10           http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
11           http://www.springframework.org/schema/beans
12           http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
13           http://www.springframework.org/schema/context
14           http://www.springframework.org/schema/context/spring-context-3.1.xsd
15           http://www.springframework.org/schema/jee
16           http://www.springframework.org/schema/jee/spring-jee-3.1.xsd
17           http://www.springframework.org/schema/lang
18           http://www.springframework.org/schema/lang/spring-lang-3.1.xsd
19           http://www.springframework.org/schema/tx
20           http://www.springframework.org/schema/tx/spring-tx-3.1.xsd
21           http://www.springframework.org/schema/mvc
22           http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
23           http://www.springframework.org/schema/util
24           http://www.springframework.org/schema/util/spring-util-3.1.xsd">
25
26       <context:component-scan base-package="controllers" />
27
28       <mvc:annotation-driven />
29
30       <bean id="viewResolver"
31           class="org.springframework.web.servlet.view.InternalResourceViewResolver">
32           <property name="viewClass"
33               value="org.springframework.web.servlet.view.JstlView" />
34           <property name="prefix" value="/WEB-INF/jsp/" />
35           <property name="suffix" value=".jsp" />
36       </bean>
37
38   </beans>
39
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xm
    <display-name>TestSpringMVCWeb</display-name>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring.xml</param-value>
    </context-param>
    <listener>
        <display-name>ContextLoaderListener</display-name>
        <listener-class>org.springframework.web.context.ContextLoaderListener</lis
    </listener>
    <servlet>
        <display-name>springmvc</display-name>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Chapter Summary

In this chapter, you learned:

- How to use Spring MVC JSP tags
- How to validate forms
- How to implement Post/Redirect/Get

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum