C
Spring Form Tag Library

The Spring Web MVC framework provides a set of tags in the form of a tag library, which is used to construct views (web pages). The Spring Web MVC integrates Spring's form tag library. Spring's form tag accesses to the command object, and also it refers to the data our Spring controller deals with. A `Command` object can be defined as a JavaBean that stores user input, usually entered through HTML form is called the `Command` object. The Spring form tag makes it easier to develop, maintain, and read JSPs. The Spring form tags are used to construct user interface elements such as text and buttons. Spring form tag library has a set of tags such as `<form>` and `<input>`. Each form tag provides support for the set of attributes of its corresponding HTML tag counterpart, which allows a developer to develop UI components in JSP or HTML pages.

Configuration – spring-form.tld

The Spring form tag library comes bundled in `spring-webmvc.jar`. The `spring-form.tld` is known as **Tag Library Descriptor** (`tld`) file, which is available in a web application and generates HTML tags. The Spring form tag library must be defined at the top of the JSP page. The following directive needs to be added to the top of your JSP pages, in order to use Spring form tags from this library:

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

Here, `form` is the tag name prefix, which will be used for the tags from this Spring form tag library in JSP pages.

The following table shows few important tags of the Spring form tag library:

Tag	Description
`form:form`	Generates the HTML `<form>` tag. It has the name attribute that specifies the command object that the inner tags should bind to.
`form:input`	Represents the HTML input text tag.
`form:password`	Represents the HTML input password tag.
`form:radiobutton`	Represents the HTML input radio button tag.
`form:checkbox`	Represents the HTML input checkbox tag.
`form:select`	Represents the HTML select list tag.
`form:options`	Represents the HTML options tag.
`form:errors`	Represents the HTML span tag. It also generates span tag from the error created as a result of validations.

The <form:form> tag

The `<form:form>` tag is used to generate an HTML `<form:form>` tag in any of the JSP pages of a Spring application. It is used for binding the inner tags, which means that all the other tags are the nested tags in the `<form:form>` tag. Using this `<form:form>` tag, the inner tags can access the `Command` object, which resides in the JSP's `PageContext` class.

The default value of the `commandName` attribute is `command`. The following code snippet shows how we use the `<form>` tags:

```
<form:form commandName="employee" action="saveEmployee"
method="post">
    ...
</form:form>
```

Here, the `<form:form>` tag contains two input tags (`firstName` and `lastName`) and one (**Save**) submit button. The value of the `firstName` and `lastName` can be accessed from the `employee` command object, which resides in the JSP's `PageContext` class.

The following code snippet shows the HTML code equivalent to the preceding code snippet (`<form:form>` tag):

```
<form method="POST">
      <table>
```

```
        <tr>
            <td>First Name:</td>
            <td><input name="firstName" type="text"
value="Ravi"/></td>
        </tr>
        <tr>
            <td>Last Name:</td>
            <td><input name="lastName" type="text"
value="Soni"/></td>
        </tr>
        <tr>
            <td colspan="2">
              <input type="submit" value="Save" />
            </td>
        </tr>
     </table>
</form>
```

The commandName attribute

The `commandName` attribute is the most important attribute in the form tag, which specifies the model attribute name that contains a backing object and the properties of this object will be used to populate the generated form. The `commandName` attribute of the `<form>` tag can be omitted for the default value that is `command`.

The <form:input>tag

The `<form:input>` tag is used for entering the text by the user in any of the JSP pages of the Spring web application. The following code snippet shows the use of the `<form:input>` tag in JSP pages:

```
<form:form>
    <table>
        <tr>
            <td>First Name:</td>
            <td><form:input path="firstName" /></td>
          </tr>
          <tr>
            <td>Last Name:</td>
            <td><form:input path="lastName" /></td>
          </tr>
          <tr>
            <td colspan="2">
```

```
                        <input type="submit" value="Save" />
                </td>
            </tr>
        </table>
    </form:form>
```

The path attribute

The `<form:input>` tag renders an HTML `<input type="text"/>` element. The `path` attribute is the most important attribute of the input tag. This `path` attribute binds the input field to the form-backing object's property.

Let's take an example, if `employee` is assigned as `commandNamed` attribute of the enclosing `<form/>` tag, then the `path` attribute of the input tag will be given as `firstName` or `lastName`. It should be noted that the `Employee` class contains getter and setter for `firstName` and `lastName` properties.

The <form:checkbox> tag

The `<form:checkbox>` tag is same as the HTML `<input>` tag, which is of the checkbox type.

```
<form:checkbox path="skills" value="Excel" label="Excel"/>
<form:checkbox path="skills" value="Word" label="Word"/>
<form:checkbox path="skills" value="Powerpoint"
label="Powerpoint"/>
```

In the preceding code snippet, the `Employee` class property called skills is used in the checkbox option. If the skills checkbox is checked, the `Employee` class's skills property is set accordingly.

The <form:radiobutton> tag

The `<form:radiobutton>` tag is used to represent the HTML `<input>` tag with the `radio` type in any of the JSP pages of a Spring web application. It is used when there are many tag instances having the same property with different values, and only one radio value can be selected at a time. The following code snippet shows how we use the `<form:radiobutton>` tag in JSP pages:

```
<tr>
    <td>Gender:</td>
    <td>
```

```
          Male: <form:radiobutton path="gender" value="male"
label="male"/>
         <br/>
        Female: <form:radiobutton path="gender" value="female"
label="female"/>
   </td>
</tr>
```

The <form:password> tag

The `<form:password>` tag is used to represent the HTML `<input>` tag of the `password` type, in any of the JSP pages of a Spring web application. By default, the browser does not show the value of the password field. We can show the value of the password field by setting the `showPassword` attribute to `true`. The following code snippet shows how we use the `<form:password>` tag in the JSP page:

```
<tr>
    <td>password :</td>
    <td>
          <form:password path="password" />
   </td>
</tr>
```

The <form:select> tag

The `<form:select>` tag is used to represent an HTML `<select>` tag in any of the JSP pages of a Spring application. We use the `<form:select>` tag for binding the selected option with its value. The `<form:option>` tag is the nested tag in the `<form:select>` tag. The following code snippet shows how we use the `<select>` tag in the JSP pages:

```
<tr>
    <td>Department</td>
    <td>
          <form:select path="department" items="${departmentMap}"
/>
   </td>
</tr>
```

The <form:option> tag

The `<form:option>` tag is used to represent an HTML `<option>` tag in any of the JSP pages of a Spring application. This tag is used when we need to add all the options to be `<form:select>` tag. Here, we need to add all the options inside the `<form:select>` tag. The following code snippet shows how to use the `<option>` tag in a JSP page:

```
<tr>
    <td>Department</td>
    <td><form:select path="department">
        <form:option value="technical" label="Technical" />
        <form:option value="non_technical" label="Non Technical" />
        <form:option value="r&d" label="R & D" />
    </form:select></td>
</tr>
```

The alternate code is as follows:

```
<tr>
    <td>Department</td>
    <td><form:select path="department">
        <form:options items="${departmentMap}" />
    </form:select></td>
</tr>
```

The <form:textarea> tag

The `<form:textarea>` tag is used to represent an HTML `<textarea>` tag in any of the JSP pages of a Spring application. The following code snippet shows how to use the `<form:textarea>` tag in a JSP pages:

```
<td>Remarks :</td>
<td>
   <form:textarea path="remarks" rows="3"
cols="20"></form:textarea>
</td>
```

The <form:hidden> tag

The `<form:hidden>` tag is used to represent an HTML hidden field in a JSP page of a Spring application. The following code snippet shows how we use the `<hidden>` tag in JSP pages:

```
<form:hidden path="empId" />
```

The <form:errors> tag

The `<form:errors>` tag is used to represent an HTML errors in a JSP of a Spring application. This tag is used for accessing the error defined in an `org.springframework.validation.Validator` interface. For example, if we want to submit a form, and find all the validator error related to the name and password fields in that form, we have to define all the validation errors related to these fields in a `Validator` class, which must implements the `Validator` interface as follows:

```
<form:form>
      <table>
          <tr>
              <td>First Name:</td>
              <td><form:input path="firstName" /></td>
              <td><form:errors path="firstName" /></td>
          </tr>
          <tr>
              <td>Last Name:</td>
              <td><form:input path="lastName" /></td>
              <td><form:errors path="lastName"  /></td>
          </tr>
          <tr>
              <td colspan="3">
                  <input type="submit" value="Save" />
              </td>
          </tr>
      </table>
  </form:form>
```

Summary

We saw the configuration of `Spring-form.tld`. We understood the `<form:form>` tag and `commandName` attributes. We also saw the `<form:input>` tag and `path` attributes. Then, we went through `<form:checkbox>`, which allows to select more than one checkbox whereas `<form:radiobutton>`. Checkbox allows only one radio button to select. Lastly, we saw `<form:select>`, `<form:textarea>`, `<form:hidden>` to hide elements and `<form:error>` for HTML errors.