



Spring MVC - View technology (JSPs) EL, JSTL and POST requests

Artur Boronat

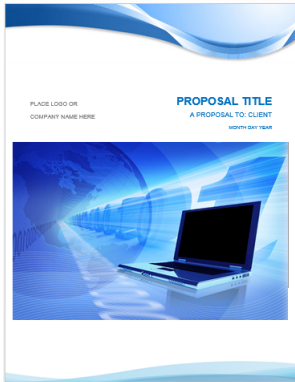


spring



Views: JSPs (JavaServer Pages)

- JSP files work as templates



- The controller chooses which template to apply by name (return value)
- The view resolver (configured in **WebConfig.java**) resolves the template:
 - instantiates the template: fills in gaps with information from model
 - generates code

Views: JSPs (Java Server Pages)

Generation of dynamic content (HTML)

- information from **model**, prepared by the **controller**
- tag libraries for controlling generation of HTML: loops, conditions
- tag libraries for forms: to post information

Ingredients

- **Expression language**: to fetch attribute values from model
- **JSTL** (JavaServer Pages Standard Tag Library): tags to define loops and conditions
- **Spring form tag library**: to design web forms that integrate well with Spring MVC

Views: Expression Language (EL)

EL

- language to evaluate expressions (returning a value)
- no loops, no conditions

How to use it

- `${expr}`: outputs the result of the expression in an HTML page
 - in view `example.jsp`: `<p>${product.getName()}</p>`
- we can refer to model attributes

```
// in the controller class
@RequestMapping(...)
public String productDetail(@ModelAttribute("product") Product product, ... ) {
    ...
    return "example"
}
```

- difference with GStrings in Groovy: the variables in expression `expr` are fetched from the **model** (as opposed to be local or global variables in the Groovy script)

Views: JSTL (JavaServer Pages Standard Tag Library)

JSTL

- collection of tags
- purpose: to program UI logic (how HTML is generated)

Tag lib directive

- added at the beginning of a JSP file
- to enable using tags from a tag library
- specifies the **URI** of the library (identifier for the library)
- **prefix** to be appended to tags within the library in order to use them

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

- to use a tag

```
<p><c:out value="Hello, World!"/></p>
```

Views: JSTL (common core tags)

c:if

- evaluates an expression and displays its body content only if the expression evaluates to true

```
<c:if test="${product.getPrice() > 2000}">
  <p>Expensive product is: <c:out value="${product.getName()}" /></p>
</c:if>
```

c:choose c:when c:otherwise

- choice between a number of alternatives (like Java switch command)

```
<c:choose>
  <c:when test="${product.getPrice() <= 10}">
    Cheap product.
  </c:when>
  <c:when test="${product.getPrice() >= 100}">
    Luxury product.
  </c:when>
  <c:otherwise>
    Affordable product.
  </c:otherwise>
</c:choose>
```

Views: JSTL (common core tags)

c:forEach

- to implement loops

```
<table>
<c:forEach items="${productList}" var="product">
<tr>
  <td><c:out value="${product.getId()}" /></td>
  <td><c:out value="${product.getName()}" /></td>
  <td><c:out value="${product.getDescription()}" /></td>
  <td><c:out value="${product.getPrice()}" /></td>
  <td><a href="/product/productDetail?productId=${product.getId()}">Edit</a></td>
  <td><a href="/product/delete?productId=${product.getId()}">Delete</a></td>
</tr>
</c:forEach>
</table>
```

spring-form tag library

- tags for including web forms in a web page
- integrate well with Spring MVC

Tag lib directive

- added at the beginning of a JSP file

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


Views: spring-form (common tags)

form:form

- **http request** for submitting the form:
 - action (URL)
 - HTTP method (POST)
- **modelAttribute**: object whose attributes can be used from a form (must be a model attribute) - used to be called **commandName**

```
<form:form method="POST" modelAttribute="product" action="/product/add">
  <table>
    <tr>
      <td><form:label path="id">Id</form:label></td>
      <td><form:input path="id" readonly="true"/></td>
    <tr>
      <td><form:label path="name">Name</form:label></td>
      <td><form:input path="name" /></td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="submit" value="Submit"/>
      </td>
    </tr>
  </table>
</form:form>
```

Views: spring-form (common tags)

input tag

- renders the value of the object attribute defined as path using type='text' by default.
- other HTML5-specific types: 'email', 'tel', 'date', etc.

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

hidden tag

- tag renders an HTML 'input' tag with type 'hidden' using the bound value

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

Views: spring-form (common tags)

select tag

- renders a drop-down list
- the option tag used to define the different elements in the list

```
<tr>
  <td>House:</td>
  <td>
    <form:select path="house">
      <form:option value="Gryffindor"/>
      <form:option value="Hufflepuff"/>
      <form:option value="Ravenclaw"/>
      <form:option value="Slytherin"/>
    </form:select>
  </td>
</tr>
```

Responsibility: HTTP request handling

- links a HTTP request to a method with an annotation `@RequestMapping`
- method parameters: get user input
- method body: population of the model
 - business logic (access to database, computations, etc.)
 - determines view
 - interprets exceptions arisen from business logic
- return value: view name

Controller: handling HTTP requests (POST)

- RequestMapping:
 - **value**: http URL (**form action**)
 - **method**: http method (POST)
- Controller method:
 - **@ModelAttribute**: object that is modified by the form

```
@RequestMapping(value = "/add", method = RequestMethod.POST)
public String productMaster(@ModelAttribute("product") Product product) { ... }
```