JSP Tag Libraries

Actions Supported by All the Laws of Nature

Main point 1 Preview

The JSP Standard Tag Library provides convenient action tags for many common operations on a JSP page. JSTL combined with the EL will satisfy most JSP needs.

Science of Consciousness: The TM Technique is a simple repeatable procedure that increases our ability to act effectively and fulfill any need.

“Scripting considered harmful”

- JSP scripting originated in early days of web apps
- most JSP scripting no longer used
  - Action elements and EL concepts used in JSF and similar frameworks
- might see in legacy code
- might see something similar in other frameworks
  - PHP
  - ASP.net
  - ASPMVC.net
  - …?
- scripting part of “Model 1 JSP architecture”
- see “memo” p314
  - circa 2000

major disadvantages of scriptlets are:

- Reusability: you can't reuse scriptlets.
- Replaceability: you can't make scriptlets abstract.
- OO-ability: you can't make use of inheritance/composition.
- Debuggability: if scriptlet throws an exception halfway, all you get is a blank page.
- Testability: scriptlets are not unit-testable.
- Maintainability: per saldo more time is needed to maintain mingled/cluttered/duplicated code logic.
- Sun Oracle itself also recommends in the JSP coding conventions to avoid use of scriptlets whenever the same functionality is possible by (tag) classes.
- StackOverflow: how to avoid Java code in JSP files? (2010)

Recall: Why You Shouldn't Use Java in a JSP

- You can do almost everything in JSP that you can do using Java.
  - But that does not mean you should do it.
- JSP is a technology that was designed for the *presentation layer*, also known as the view.
  - In most organizations, user interface developers are responsible for creating presentation layer.
    - These developers rarely have experience writing Java code, and providing them with the ability can be dangerous.
- In a well-structured, cleanly coded application, the presentation layer is separated from the business logic, which is likewise separated from the data persistence layer.
- It's actually possible to create JSPs that display dynamic content without single line of Java inside the JSP. That's our mission!!!

Recall: Forwarding a Request from a Servlet to a JSP

- A typical pattern when combining Servlets and JSPs is to have the Servlet accept the request, do any business logic processing and data storage or retrieval necessary, prepare a model that can easily be used in JSP, and then forward request to the JSP.

```
request.getRequestDispatcher("/WEB-INF/jsp/welcome.jsp")
.forward(request, response);
```

JSP Actions

- JSP actions are xml elements or tags that the container executes
  - scripting often not suitable for HTML content developers
  - JSP Actions are predefined HTML-like elements for common processing tasks such as iteration, conditionals, database access, etc.

- JSP Standard Actions
  - “Standard” in sense that are included with every JSP implementation—part of JSP specification
  - Examples

```
<jsp:useBean id="connection“  class="com.myco.myapp.Connection“ scope=“session”>
     <jsp:setProperty name="connection" property="timeout" value="33">
</jsp:useBean>

<jsp:forward page="error.jsp" />

<jsp:include page="hello.jsp"/>
```

- JSTL
  - Java Standard Tag Library
  - many libraries of JSP actions (“tags”)
  - JSTL is one tag library that is widely used and has this name
  - not part of the base JSP implementation, must be added to an app as a library

Using JSTL

- There are five tag libraries in the Java Standard Tag Library specification:
  - Core (`c`)
  - Formatting (`fmt`)
  - Functions (`fn`)
  - SQL (`sql`)
  - XML (`x`)
- Examples of tags from the Core JSTL Library
  - `c:set(set value of a variable)`
  - `c:out`
  - `c:if(conditional)`
  - `c:choose`
  - `c:forEach`
- Many of these make simple use of the EL.

Quick EL Review

- An expression `${person.age}` retrieves an attribute object using “person” as a key (and prints to the page) something like: xyz`.getAge()`, where xyz is an instance of a Java object having an age property

- An expression `${expr}` prints the value of `expr` to the page, where `expr` is a Java expression using Java arithmetic or logical operators

c:out to avoid XSS attack

```
<p>The person's name is <c:out value="${person.name}" /></p>
<p>The person's name is ${person.name}</p>
```

- c:out escapes HTML characters
  - if person.name = <script>alert("You are hacked!")</script>
  - the script will be executed in the second case, but not when using c:out

JSTL example with body

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html><head><title>Weather Page</title></head>
<body>
<%String[][] data = {{"Nov 6", "32", "26"},{"Nov 7", "32", "26"},{"Nov 8", "32", "26"}};
request.setAttribute(“temperatures", data);%>

<table>
<tr><th>DATE</th><th>HIGH</th><th>LOW</th></tr>
<c:forEach var=“daily" items="${temperatures}">
   <tr>
      <td>${daily[0]}</td><td>${daily[1]}</td><td>${daily[2]}</td>
   </tr>
</c:forEach>
</table></body></html>
```

DATE	HIGH	LOW
Nov 6	32℃	26℃
Nov 7	32℃	26℃
Nov 8	32℃	26℃

JSTL demo

- visually preview the 6 demo steps
- what HTTP message will step 6 generate and what will happen on the server
- implement and run the demo
- if you finish quickly, try rewriting the JSP page using scripting instead of JSTL

Using JSTL

The JSTL library provides 5 kinds of tags, each having a different (standard) prefix. You “import” a library by placing a “taglib” directive at the top of your jsp page. Here are the choices:

- Core tags: (if/else, loops, choose …)

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

- Function tags: (standard Java string manipulation)

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

Using JSTL (continued)

- Format Tags (format numbers, dates..)

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

- SQL Tags (set data source, perform queries)

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

- XML Tags (for navigating/parsing/working with XML files)

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

Main point 1

The JSP Standard Tag Library provides convenient action tags for many common operations on a JSP page. JSTL combined with the EL will satisfy most JSP needs.

Main point 2 Preview

Developers can create custom tags for JSP that use a Java tag handler to provide any desired functionality in a simple tag definition for JSP authors.

Science of Consciousness: Knowledge of custom tags gives developers a means to create any JSP tag functionality. Knowledge and experience of the unified field gives us access to the source of all possibilities.

Custom tags

- JSTL is a standard library of JSP actions, but JSP allows developers to create their own actions
- component development creates custom functionality that can be packaged and reused by content developers
- almost every server side web app framework relies heavily on the use of such components
- key steps
  - define a tag including attributes and body
  - write a Tag Library Descriptor (TLD) that the container will read
  - write a tag handler class that implements the tag functionality
  - use the tag on a JSP page and link it to the tag descriptor

A simple custom tag

- define the tag

```
<aspx:Label foreColor='red' text='Hello'/>
```

will generate the following HTML:

```
<span style='color:red'>Hello</span>
```

- define a Tag Library Descriptor (TLD) for the tag

```
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
<uri>/WEB-INF/tlds/TldDemo</uri>
 <tag>
    <description>Generates a label</description>
    <name>Label</name>
    <tag-class>net.mumde.cs545.Label</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>foreColor</name>
      <required>false</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
    <attribute>
      <name>text</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag></taglib>
```

Write a tag handler class

```
public class Label extends SimpleTagSupport

{String foreColor;
 String text;
 public void doTag() throws JspException, IOException    //render custom tag
       {
             JspWriter out = getJspContext().getOut();
             if (foreColor != null)
                    out.write(String.format("<span style='color:%s'>%s</span>", foreColor, text));
             else
                    out.write(String.format("<span>%s</span>", text));
       }
//Need a setter for each attribute of custom tag
       public void setForeColor(String foreColor)
       {     this.foreColor = foreColor;
       }
       public void setText(String text)
       {     this.text = text;
       }
 }
```

Use the tag

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix='aspx' uri='/WEB-INF/tlds/TldDemo'%>
<html>
<body>
<aspx:Label foreColor='red' text='hello'/>
</body>
</html>
```

uri is any unique string

- implement and run the custom tag demo

The Java Server Page – taglib directive

- The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.

- Example:
  - `<%@ taglib uri="…" prefix="…" %>`

- The uri attribute value is a unique string that the container will use to identify the appropriate TLD and the prefix attribute identifies the tag components on this particular JSP page

Tags with bodies

- examples of tags with bodies
  - c:forEach tag – will loop through collection and regenerate the body each time with list element inserted (HF 437)
  - c:if, c:when, c:choose all have bodies that get inserted conditionally (HF 443-444)

- need to call getJspBody().invoke(null) in the doTag() to process the body of the tag

(Recall) JSTL example with body

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html><head><title>Weather Page</title></head>
<body>
<%String[][] data = {{"Nov 6", "32", "26"},{"Nov 7", "32", "26"},{"Nov 8", "32", "26"}};
request.setAttribute(“temperatures", data);%>

<table>
<tr><th>DATE</th><th>HIGH</th><th>LOW</th></tr>
<c:forEach var=“daily" items="${temperatures}">
   <tr>
      <td>${daily[0]}</td><td>${daily[1]}</td><td>${daily[2]}</td>
   </tr>
</c:forEach>
</table></body></html>
```

DATE	HIGH	LOW
Nov 6	32℃	26℃
Nov 7	32℃	26℃
Nov 8	32℃	26℃

Tag handler for c:forEach

```
public class forEachTagHandler extends SimpleTagSupport {
   private String[][] items;
   private String var;

   public void doTag() throws JspException, IOException  {
      JspWriter out = getJspContext().getOut();
      int rows = items.length;
      /* invoke body on each row of the input table */
      for (int i=0; i< rows; i++) {
         /* set an attribute in this page (name=var, value=items[i]) */
         getJspContext().setAttribute(var, items[i]);
         /* invoke the body for the next row in the table */
         getJspBody().invoke(null);
      }
   }
   public void setItems(String[][] items) { this.items = items; }
   public void setVar(String var) { this.var = var; }
}
```

Exercise for reader: create the TLD entry

Why JSP custom tags are important

- main purpose of most JSP custom tags is to provide an easy mechanism to dynamically “generate markup” for common processing tasks
- Current best practice with JSP is to use tags on pages and avoid any JSP scripting elements
- JSTL and many other libraries of custom tags have been created
- SpringMVC uses JSP tags and has its own tag library
- JSF is a component based framework that is essentially a large set of custom tags running in a special lifecycle
- Client side JavaScript frameworks such as React and Angular have their own markup generation components

- reuse through software components is an important general principle of software engineering. As web applications gain sophistication and complexity, component reuse becomes increasingly important.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Actions Supported by All the Laws of nature

1. Custom tags are easy for JSP page authors to use.

2. Java developers can create custom tags for any functionality.

---

3. **Transcendental consciousness** is the experience of the home of all the laws of nature.

4. **Impulses within the transcendental field:** Thoughts that arise from this transcendental field will be naturally in accord with all the laws of nature because this transcendental field is the unified field from which the laws of nature arise.

5. **Wholeness moving within itself:** A developer who understands the basic coding mechanisms underlying custom tags will feel a deep level of comfort and connection with JSP pages and web application technologies and frameworks that rely on them. In a similar manner, in unity consciousness we appreciate that the pure consciousness we experience as our deepest Self is also the same unified field of consciousness that is expressed as the rest of the manifest world and feel a deep level of comfort and connection with that.