JSP TAG LIBRARIES

Actions Supported by All the Laws of Nature

“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

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

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℃

Lots of JSTL tags

```
<c:choose>
  <c:when test="${customer.category == 'trial'}" >
    ...
  </c:when>
  <c:when test="${customer.category == 'member'}" >
    ...
  </c:when>
    <c:when test="${customer.category == 'preferred'}" >
    ...
  </c:when>
  <c:otherwise>
    ...
  </c:otherwise>
</c:choose>
```

JSTL core	JSTL 1.1 core library
JSTL fmt	JSTL 1.1 i18n-capable formatting library
JSTL sql	JSTL 1.1 sql library
JSTL XML	JSTL 1.1 XML library
JSTL functions	JSTL 1.1 functions library

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

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 modern 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:

Hello

- 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>
  <short-name>tlddemo</short-name>
  <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

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
- basic concept of components for markup generation will be found in most web app frameworks
- e.g., JSF is a component based framework
  - all JSF tags will be implemented as “custom tags”
  - slightly more complex because have to satisfy requirements of the JSF framework

Review

- unity chart
- week 1 review
- lab 6 – state management exercise