JSP Tag Libraries

Actions Supported by All the Laws of Nature

JSP Techniques

We discuss the following techniques

1. Standard custom tags for JSPs: JSTL
2. The JSP custom tags api

JSTL – JSP Standard Tag Library

- Provides new tags for JSPs that reduce scripting
- They use the custom tag api, but have become a standard library, essentially a part of JSP language

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: (can do if, if/else, loops…)

```
<%@ 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/core" %>
```

(continued)

- Format Tags (format numbers, dates..)

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

- 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" %>
```

(continued)

- Head First, p. 475 lists all the available JSTL library tags.
- 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 ${expr} prints expr to the page
- An expression ${person.name} evaluates (and prints to the page) something like: person.getName(), where person is an instance of a Person bean having a *name* variable.
- To test whether two objects are equal, EL uses **eq** for equals and **ne** for not equals

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°C	26°C
Nov 7	32°C	26°C
Nov 8	32°C	26°C

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

JSP Custom Tag Libraries

- Tag handler class
- Descriptor file
- JSP taglib directive
- Example: Print a label with font and color:
  label.jsp

Components of a tag library

- Tag library descriptor document
- Tag handler class
- Java Server Pages that use the tag

Tag Library Descriptor

- An xml document that contains information about a library as a whole and about each tag contained in the library.
- Used by a web container to validate the tags and by JSP page development tools.
- Example: label.tld

Tag Library Descriptor

```
<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.mum.waa.custom.tag.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>
```

Tag handler class

- Extends the tag support framework class – javax.servlet.jsp.tagext.TagSupport

Tag handler class

```
public class Label extends SimpleTagSupport{
  String foreColor;
  String text;
   //render custom tag
  public void doTag() throws JspException, IOException  {
     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;
   }
}
```

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 resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.

The Java Server Page

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

Why use custom tags?

- To provide an easy mechanism to dynamically "generate markup" for common processing tasks.
- e.g., JSF is a component based framework. All JSF tags will be implemented as "custom tags". More complex because they have to satisfy requirements of the JSF framework.