Java Server Pages (JSP)

Basics

- Whereas servlets often embed HTML in Java, JSP's embed Java in HTML
- JSP's are translated into servlets and compiled by the web server
- JSP's consist of standard HTML, scripting elements, directives, and action elements
- HTML in the JSP will produce println statements in the generated servlet

HTML in JSP

- Comments:

  <!-- *HTML Comment* --> HTML comments are passed through to the client.

  <%-- *JSP Comment* --%> JSP comments are not.
- To use <% or %> in the HTML use <\% or % \> respectively.

Scripting Elements

- Expressions: <%= *expression* %>
  - The generated servlet will evaluate the expression and insert the result in its output
- Scriptlets: <% *code* %>
  - The code will be inserted in a method invoked by the servlets service method
- Declarations: <%! *code* %>
  - The code is inserted in the servlet class outside of any methods

Alternative XML Syntax

- Expressions:

  <jsp:expression> *expression* </jsp:expression>
- Scriptlets:

  <jsp:scriptlet> *code* </jsp:scriptlet>
- Declarations:

  <jsp:declaration> *code* </jsp:declaration>

Expressions

- Evaluated at runtime (when page is requested), converted to String, and inserted in the page.
- Predefined Variables
  - **request**, the HttpServletRequest
  - **response**, the HttpServletResponse
  - **session**, the HttpSession for the request
  - **out**, a PrintWriter to write to the client
  - **application**, the ServletContext
  - **config**, the ServletConfig
  - **pageContext**, the PageContext
  - **page**, same as this

Expression Examples

```
<h2>JSP Expressions</h2>
<ul>
  <li>Current time: <%= new java.util.Date() %> </li>
  <li>Your hostname: <%= request.getRemoteHost() %></li>
  <li>Test value: <%= request.getParameter("test") %> </li>
  <li>Your session ID: <%= session.getId() %> </li>
</ul>
```

JSP Expressions

- Current time: Mon Jun 3 11:20:15 EST 2002
- Your hostname: moosehead.cis.umassd.edu
- Test value: some data
- Your session ID: YTGFREDKI8P6HF43WAXDLKUY

Scriptlets

- Have access to the same predefined variables as expressions.
- Allow the inclusion of arbitrary code in the generated servlet.
- May be interspersed with plain HTML and expressions.
- Can make parts of page conditional, or generated repeatedly in a loop.

Scriplet Example (1)

```
<% if (Math.random() < 0.5) {  %>
    Have a <b>nice</b> day!
<% } else { %>
    Have a <b>lousy</b> day!
<% } %>
```

JSP Code

```
if (Math.random() < 0.5) {
    out.println("Have a <b>nice</b> day!");
} else {
    out.println("Have a <b>lousy</b> day!");
}
```

Generated Servlet Code

Scriptlet Example (2.1)

```
<table border="1">
<tr><th>A</th><th>B</th></tr>
<% while (results.next()) { %>
  <tr>
  <td><%= results.getString(1) %></td>
  <td><%= results.getString(2) %></td>
  </tr>
<% } %>
</table>
```

Scriptlet Example (2.2)

```
out.println("<table border=\"1\">");
out.println("<tr><th>A</th><th>B</th></tr>");
while (results.next()) {
    out.println("<tr>");
    out.print("<td>");
   out.print((results.getString(1)).toString());
   out.println("</td>");
    ...
```

Declarations

- Example

  ```
  <%! private int count = 0; %>
  This page has been accessed
  <%= ++count %> times since the last server reboot.
  ```

- Note that count need not be static, since there is normally only one instance of each servlet (possibly accessed simultaneously by multiple threads).

Page Directives

<%@ page attribute="value" … %>

- import attribute
  - classes and packages used on the page
  - may appear multiple times
  - <%@ page import="java.util.*" %>
- contentType attribute
  - e.g. "text/html" (the default for JSP's)
- isThreadSafe attribute
  - values are "true" (the default) or "false"
  - if false, the servlet will implement SingleThreadModel

Page Directives (2)

- session attribute
  - values are "true" (the default) or "false"
  - if true the page uses HTTP sessions, creating one if it doesn't already exist
- buffer attribute
  - the size of the buffer in KB used by the *out* variable or "none" to turn off buffering
- autoflush attribute
  - "true" (default) or "false"
  - if true, the buffer will be flushed when it fills up
  - if false, a buffer overflow exception may be raised

Page Directives (3)

- extends attribute
  - the superclass of the generated servlet
  - <%@ page extends="*package.class*" %>
  - caution: the server may already use a custom class
- info attribute
  - a string that can be retrieved from the servlet with the *getServletInfo* method
- errorPage attribute
  - URL of page to process any exceptions that are thrown and not caught on the current page
  - The exception will be available on the error page through the *exception* variable.

Page Directives (4)

- isErrorPage attribute
  - "true" or "false" (default)
  - if true may act as error page for some other page
- language attribute
  - only legal value is "java" (the default)

Include Directives

<%@ include file="*relativeURL*" %>

- The included file may be either static (*.html) or dynamic (*.jsp).
- The file is included when the JSP is translated into a servlet, i.e. when the page is first requested.
- When a dynamic file is included, it is the actual file itself (not its output) that is included.
- If an included file is modified, the modification dates of all files that include it must be updated!

Taglib Directives

<%@ taglib uri="*tagLibURI*" prefix="*tagPrefix*" %>

- Introduces a tag library and defines a prefix for custom tags used on the page.
- More later

Alternative XML Directive Syntax

<jsp:directive.*directiveType attribute*="*value*" />

- For example:

  <jsp:directive.page import="java.util.*" />

Action Elements

- jsp:include
- jsp:forward
- jsp:useBean
- jsp:getProperty
- jsp:setProperty
- jsp:plugin

jsp:include

<jsp:include page="*relativeURL*" flush="*true*" />

- If the included page is dynamic (JSP or servlet) the *output* of the page rather than the page itself is included.
- The inclusion takes place each time the main page is requested.
- Two required parameters:
  - page (the url of the included page)
  - flush (*must* have the value "true")

jsp:forward

<jsp:forward page="*relativeURL*" />

<jsp:forward page="*relativeURL*" >
    <jsp:param name="*pName*" value="*pValue*">
    …
</jsp:forward>

- Forwards the request to another page, optionally specifying additional parameters and values.
- The value of page may be an expression, so the value may be conditional (e.g. on the value of some parameter)

jsp:forward (2)

```
<%
String id = request.getParameter("id");
String destination;
if (id.startsWith("a"))
   destination = "adminLogin.jsp";
else
   destination = "userLogin.jsp";
%>
<jsp:forward page="<%= destination %>" />
```

JavaBeans

- A bean class must have a default (zero argument) constructor.
- A bean class may have *properties* and *methods* that can be accessed in JSP.
- A property is a private field with public get and set methods.
- A property xxx has accessor methods:
  - public void setXxx(type x)
  - public type getXxx() // if type is not boolean
  - public type isXxx() // if type is boolean

jsp:useBean

<jsp:useBean id="*name*" class="*package.Class*" />

- The package or class needs to be imported in a page directive.
- Creates an object of the specified class and binds it to a variable with the specified id (unless a bean with that id already exists within the same scope).
- Optional scope attribute can be used to share beans with other JSP's and servlets.

Bean scope

- scope="page" (default)
  - Bean is available only on the same page for the duration of the current request
- scope="request"
  - Bean is stored in the ServletRequest for the duration of the current request. Could include other pages (e.g. if the request is forwarded).
- scope="session"
  - Bean is stored in the HttpSession and is available to all servlets and JSP's for the duration of the current session.
- scope="application"
  - Bean is stored in the ServletContext and is available to all servlets and JSP's in the same webapp

Accessing bean properties

```
<jsp:getProperty name="beanID"
   property="propertyName" />
<%= beanID.getPropertyName() %>
```

```
<jsp:setProperty name="beanID"
   property="propertyName" value="string" />
<% beanID.setPropertyName("string"); %>
```

- If the property has a type other than String, jsp:setProperty will perform the type conversion.

Setting properties with parameters

```
<jsp:setProperty name="beanID"
   property="propertyName"
   param="paramName" />
```

- If the parameter has the same name as the property, the param attribute may be omitted.
- Use "*" for property attribute to set *all* properties to values of identically named parameters.

Conditional bean initialization

<jsp:useBean … >
  statements
</jsp:useBean>

- The statements are executed only if a new bean was created, not if an existing bean is used.
- Useful if a bean is shared by several pages and it is unknown which page will be accessed first.

Using Tag Libraries

<%@ taglib uri="*TLDUrl*" prefix="*tagPrefix*" %>

- The uri attribute gives the URL of a Tag Library Descriptor file for the tag library.
- The prefix attribute specifies a prefix to be used for tags defined in the library.

  <prefix:tagname … />

JSP Standard Tag Library (JSTL)

- JSTL has multiple tag library descriptors (TLDs) to clearly show the functional areas it covers and give each area its own namespace.
- The JSTL tag libraries come in two versions which differ only in the way they support the use of expressions for attribute values.
  - In the JSTL-RT tag library, expressions are specified in the page's scripting language.
  - In the JSTL-EL tag library, expressions are specified in the JSTL expression language

JSTL Expression Language

- Allows simplified syntax for accessing attributes:

  ${aName} instead of
  <%= pageContext.getAttribute("aName") %>

  ${aName.foo.bar} instead of
  <%= aName.getFoo().getBar() %>
- Allows easy access to application data without using scriptlets

```
<c:if test="${sessionScope.cart.numberOfItems > 0}">
  ...
</c:if>
```

RT versus EL

- The RT library is usually better when the JSP page directly computes all values.
- The EL library is better when the JSP page uses only values computed by servlets (e.g. when using the MVC architecture).
- In between these extremes, it may be mostly a matter of taste.

Core JSTL Tags

TLD's	Functions	Tags
c.tld c-rt.tld	Expression Language Support	catch out remove set
	Flow Control	choose when otherwise forEach forTokens if
	URL Management	import paramInedirect param url param

XML JSTL Tags

TLD's	Functions	Tags
x.tld x-rt.tld	Core	out parse set
	Flow Control	choose when otherwise forEach if
	Transformation	transform param

Internationalization JSTL Tags

TLD's	Functions	Tags
fmt.tld fmt-rt.tld	Locale	setLocale
	Message Formatting	bundle message param setBundle
	Number and Date Formatting	formatNumber formatDate parseDate parseNumber setTimeZone timeZone

Database JSTL Tags

TLD's	Functions	Tags
sql.tld sql-rt.tld		setDataSource
	SQL	query dateParam param transaction update dateParam param

Flow Control Examples

- <c:if test="${x < 5}">
  …
  </c:if>
- <c:choose>
  <c:when test="${x < 5}">
  …
  </c:when>
  <c:when test="${x == 5}">
  …
  </c:when>
  <c:otherwise>
  …
  </c:otherwise>
  </c:choose>

Flow Control Examples (2)

```
<table>
  <c:forEach var="customer" items="${customers}">
    <tr><td><c:out value="${customer}"/></td></tr>
  </c:forEach>
</table>
```

Database Example

```
<sql:setDataSource
   url="jdbcURL"
   driver="driverClassName"
   user="username"
   password="password" />

<sql:query var="customers">
   SELECT * FROM customers
      WHERE country = 'China'
      ORDER BY lastname
</sql:query>
```

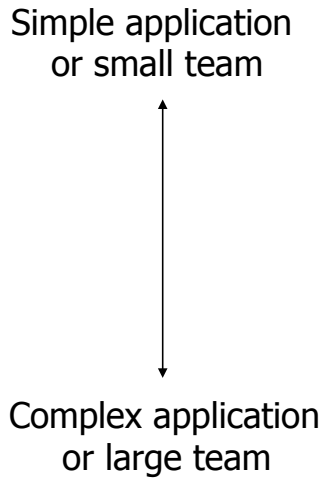
Database Example (2)

```
<table>
  <c:forEach var="row" items="$
   {customers.rows}">
  <tr>
    <td><c:out value="${row.lastName}"/></td>
    <td><c:out value="${row.firstName}"/></td>
    <td><c:out value="${row.address}"/></td>
  </tr>
  </c:forEach>
</table>
```

Defining Custom Tag Libraries

- Custom tags can manipulate JSP content; beans cannot.
- Complex operations can be reduced to simpler form with custom tags than with beans.
- Custom tags are more work to setup than beans.
- Custom tags are not supported in JSP 1.0 but beans are.

Development Approaches



- Place all Java directly in JSP or all HTML in servlet code.
- Call Java code in separate utility classes from scriptlets.
- Use Java Beans.
- Use custom tags.
- Use the MVC architecture.

MVC Architecture

- Keep the model, the control, and the views separate.
  - Model: Database and servlets that capture business logic.
  - Control: Servlets that capture application logic.
  - View: JSP's that capture presentation logic.
- A servlet may respond to original request, look up or compute data, store results in beans and forward to JSP to present results.