Module 6

JSP Specification and Syntax

Course Objectives

- After completing this module, you should be able to:
  - Write JSP pages using the core JSP syntax
  - Directives
  - Declarations
  - Scriptlets
  - Expressions
  - Describe the general content of a JSP page
  - Describe how to generate XML from a JSP document
  - Identify the XML syntax for JSP directives, declarations, scriptlets and expressions
  - Describe JSP interaction with HTML, servlets and other JSP pages

JSP 2.1 specification

- Java Server Pages 2.1 provides two syntax types for JSP files:
  - JSP Page syntax
  - JSP Document (XML) syntax
- JSP Page syntax is discussed in this presentation. JSP Document syntax is discussed later.

JSP syntax elements

- JSP elements fall into four groups:
  - Directives
  - Scripting
    - Declarations
    - Expressions
    - Scriptlets
  - Comments
  - Actions
- Directives and scripting have both non-XML syntax and XML syntax
- Actions are *only* written in XML syntax

JSP directives

- JSP directives are instructions processed by the JSP engine when the page is compiled into a servlet
  - `<%@ directive {attribute="value"}* %>`

- JSP technology currently defines page, include, and taglib directives
  - `<%@ page language="java" %>`
  - `<%@ include file="companyBanner.html"%>`
  - `<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>`

The JSP page directive

- Defines page-specific attributes
  - Apply to complete translation unit
- Syntax: **<%@ page attribute* %>**
  - where attributes may include any of:
    - language="scriptingLanguage"
    - extends="className"
    - import="importList"
    - session="true|false"
    - buffer="none|sizekb"
    - autoFlush="true|false"
    - info="info_text"
    - isErrorPage="true|false"
    - contentType="ctinfo"
    - pageEncoding="peinfo"
    - isELIgnored="true|false"

JSP page directives: import

- Comma-separated list of Java language package names or class names that the servlet imports
- Can be specified multiple times within a JSP file to import different packages

```
<%@ page import ="java.io.*,java.util.Hashtable" ... %>
```

JSP page directives: session

- session
  - If true, this page participates in an HTTP session. The implicit script variable session (of type javax.servlet.http.HttpSession) is the session for the page
  - If false, the session variable is not available and any reference to it results in a fatal translation error
  - Defaults to true

```
<%@ page session="true" ... %>
```

JSP page directives: buffer

- buffer
  - Specifies the buffering mode for the out JspWriter
  - If buffer="none", there is no buffering, and all of the output is written directly to the ServletResponse object's PrintWriter
  - If a buffer size is provided, all output is written to a buffer with a size not less than the specified value
  - If no buffer parameter is specified, the default is a buffered JspWriter with a buffer size not less than 8 KB
  - See also autoFlush

```
<%@ page buffer="12kb" ... %>
```

JSP page directives: autoFlush

- autoFlush
  - For buffered output
  - If true, the output is flushed automatically when the buffer is full
  - If false, an exception is thrown if the buffer is full
  - It is illegal to specify autoFlush="false" when buffer="none"
  - Defaults to true

```
<%@ page autoFlush="true" ... %>
```

JSP page directives: isErrorPage

- isErrorPage
  - Indicates that the JSP page is intended as the URL target of another JSP page's errorPage
  - If true, the variable exception is defined, and is a reference to the Throwable object of the JSP page in error
  - Defaults to false

```
<%@ page isErrorPage="true" ... %>
```

JSP page directives: errorPage

- errorPage
  - Defines a URL to a JSP page that is invoked if a throwable object is thrown but not caught by this JSP
  - The default URL is implementation-dependent

```
<%@ page errorPage="/oops.jsp" ... %>
```

JSP page directives:contentType

- contentType
  - The MIME type of the generated response. This information is used to generate the response header. You can optionally use this directive to specify the character set in which the page is to be encoded.
  - Syntax:
    - contentType="TYPE" OR
    - contentType="TYPE;charset=CHARSET"
  - Default values for TYPE= "text/html" and CHARSET="ISO-8859-1"

```
<%@ page contentType = "text/html" ...%>
```

OR

```
<%@ page contentType = "text/html;charset=iso-8859-1" ...%>
```

The <%@ include %> and <%@ taglib %> directives

- Include is used to substitute text or code at translation (compile) time

```
<%@ include file="relativeURLspec" %>
```

- Taglib is used to declare a tag library containing custom tags

```
<%@ taglib uri="tagLibraryURI“ prefix="tagPrefix"%>
```

JSP scripting (1 of 4)

- Declarations
  - Used to declare instance and class variables and methods in the scripting language used in a JSP page

```
<%! declaration %>
<%!
   private int getDateCount =0;
      private String getDate (GregorianCalendar gc1)
      {...method body here...}
 %>
```

  - `jspInit` and `jspDestroy` methods may be defined here

JSP scripting (2 of 4)

- Scriptlets (inline Java code)
  - Java code makes up the body of the generated method

```
<% valid_code_fragment %>

<%if(Calendar.getInstance().
        get(Calendar.AM_PM)==Calendar.AM){%>

        How are you this morning?

<%}else{%>

        How are you this afternoon?

<%}%>
```

JSP scripting (3 of 4)

- Comments
  - Similar to HTML comments
  - Stripped from the page by the JSP engine when executed

```
<%-- comment --%>

<%-- check for AM or PM --%>
```

JSP scripting (4 of 4)

- Expressions
  - The expression is evaluated
  - Result is converted to a String and displayed

```
<%= expression %>
```

  - The following example calls the incrementCounter method declared in the declarations block, and prints the result

```
<%= incrementCounter() %>
```

Variables

- Within both Scriptlets and Expressions, there are certain implicit objects available for use (without being declared first)
- Implicitly defined variables available for scripting:
  - **request**: HttpServletRequest object
  - **response**: HttpServletResponse object
  - **pageContext**: the PageContext for this JSP page
  - **session**: HttpSession object (if any)
  - **application**: the ServletContext object
  - **config**: ServletConfig object for this JSP page
  - **out**: JspWriter
  - **page**: the JSP page's implementation class processing the current request (for Java, page is a synonym for this)
  - **exception**: the throwable object passed to this error page

Scope factors

Scope	Type	Variable	Factor
page	PageContext	pageContext	Used internally by the page compiler, and by custom tag libraries.
request	ServletRequest	request	Information relevant to a specific user for a specific HTTP request/response pair.
session	HttpSession	session	Information relevant to a specific user for a series of request/response pairs.
application	ServletContext	application	Information specific to a group of users across multiple servlets in a Web application.

Exception Handling

- You can provide your own exception handling within JSP pages
  - It may not be possible to anticipate all situations
- Use the page directive's errorPage attribute
  - Forward an uncaught exception to an error handling JSP page for processing.

```
<%@ page isErrorPage="false" errorPage="errorHandler.jsp" %>
```

  - flag errorHandler.jsp as an error processing page:

```
<%@ page isErrorPage="true" %>
```

The include and forward actions

- Interact with the RequestDispatcher mechanism
- Include

```
<jsp:include page="StdHeader.jsp" flush="true" />
```

  - Execution of current page continues after including response from target
- Forward

```
<jsp:forward page="ExtraInfo.jsp" />
```

  - Execution of current page is terminated and target resource has full control over request

JSP interactions

- You can invoke a JSP page:
  - By the URL
  - By a servlet
  - By another JSP page
- A JSP page can invoke:
  - A servlet
  - Another JSP page

Invoking a JSP by URL

- You can invoke a JSP page by URL
  - From within the <FORM> tag of a JSP or HTML page
  - From another JSP page
- To invoke a JSP page by URL, use the following syntax:

```
http://servername/path/filename.jsp
```

Calling a JSP page from a servlet

- You can call JSP pages using the same RequestDispatcher interface that you use to call servlets from servlets

```
getServletContext().
        getRequestDispatcher
                ("/pages/showResults.jsp").
                        forward(req, res);
```

Invoking a servlet from a JSP page

- You can invoke a servlet from a JSP page, either as an action on a form or directly through the jsp:include or jsp:forward tags.
  - To invoke a servlet within the HTML <FORM> tag, the syntax is as follows :

```
<FORM METHOD="POST|GET" ACTION="application_URI/Servlet_URL ">

    <!--Other tags such as text boxes and buttons go here-->

</FORM>
```

Invoking a JSP page from another JSP

- To invoke a JSP file from another JSP file, you can:
  - Specify the URL of the second JSP file on the form action attribute:

```
<form action="/MYAPP/DateDisplay.jsp">
```

  - Specify the URL of the second JSP file in an anchor tag href attribute:

```
<a href="/MYAPP/InfoDisplay.jsp"> reference-text </a>
```

Scripting example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<%TestAreas tests=
  (TestAreas)request.getAttribute("tests");%>

<TABLE cellspacing="10"><TBODY>
<TR>
<TD><FONT color="#0000cc" size="5" face="Comic Sans MS">
The following areas have tests available</FONT></TD>
</TR>

<%for (int i = 0; i < tests.getTestAreas().length; i++){%>
  <TR><TD align="center">

  <!-- The following two lines are really 1 line -->
  <A href="/Course/ExamCommand?cmd=displayTestsByArea&testArea=
  <%= tests.getTestAreas(i).getKey()%>">

  <IMG border="0"
   src="<%= tests.getTestAreas(i).getImageFileWithPath() %>">
  </A>
  </TD></TR>
<%}%>

</TBODY> </TABLE> </BODY>
```

Scripting is possible (but not encouraged) in JSP pages. It will be replaced in subsequent modules.

JSP actions

- Actions perform a variety of functions, and extend the capabilities of JSP
  - Standard
    - <jsp:useBean>
    - <jsp:setProperty>
    - <jsp:getProperty>
    - <jsp:include>
    - <jsp:forward>
    - <jsp:param>
    - <jsp:plugin> –
      - <jsp:params>
      - <jsp:fallback>
  - Custom
    - Introduced via the taglib directive

JSP document syntax

- JSP document (XML) syntax provides the same functionality as JSP page syntax with added benefits:
  - JSP documents can be manipulated by XML-aware tools
  - The XML view of a JSP document can be used to validate the document
  - A JSP document can be generated from a textual representation by applying an XML transformation
- JSP documents are namespace-aware XML documents
  - XML namespaces identify the core syntax
- The JSP document and JSP page syntax cannot be mixed in one file
- A JSP file in one syntax can include or forward to a file in the other syntax

XML tags (1 of 2)

- The JSP specification provides special tags for XML documents:
  - XML Document root tag

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
        ...
</jsp:root>
```

  - Directives

```
<jsp:directive.directive { attr="value" }* />
```

    Declarations

```
<jsp:declaration>
        ...
</jsp:declaration>
```

XML Tags (2 of 2)

- Scriptlet

```
<jsp:scriptlet>
    code fragment goes here
</jsp:scriptlet>
```

- Expressions

```
<jsp:expression>
    expression goes here
</jsp:expression>
```

Document versus page syntax (1 of 2)

XML JSP Syntax	Traditional JSP Syntax
<jsp:root xmlns:jsp="dtd"> remainder of JSP page </jsp:root>	None

<jsp:directive.include
  file="copyright.html" />
	<%@ include
	  file="copyright.html" %>

<jsp:declaration>
private int getDateCount =0;
private String getDate
  (GregorianCalendar gc1)
{...method body here...}
</jsp:declaration>
	<%!
	private int getDateCount =0;
	private String getDate
	  (GregorianCalendar gc1)
	{...method body here...}
	%>

Document versus page syntax (2 of 2)

XML JSP Syntax	Traditional JSP Syntax

How are you this
<jsp:scriptlet>
if (isAM) {
  out.print(" morning?");
} else {
  out.print(" afternoon?");
}
</jsp:scriptlet>
	How are you this
	<%
	if (isAM) {
	  out.print(" morning?");
	} else {
	  out.print(" afternoon?");
	}
	%>

<jsp:expression> customer.getFirstName() </jsp:expression>	<%= customer.getFirstName() %>

JSP issues

- Advantages:
  - Can build and maintain presentation using page authoring tools
  - Full support for Java on the server-side
- Disadvantages:
  - Excessive Java code pollutes HTML
  - Poor separation of responsibility
  - (.jsp owned by both Page Designer and Logic Developer)

Checkpoint

1. What is a JSP page composed of?
2. What executes when a JSP page is invoked?
3. Which JSP scripting statement does not allow a closing semicolon?
4. Why is the <jsp:root> tag needed?

Module Summary

- This unit covered the following topics:
  - Write JSP pages using the core JSP syntax
    - Directives
    - Declarations
    - Scriptlets
    - Expressions
  - Describe the general content of a JSP page
  - Discuss how to generate XML from a JSP page
  - Identify the XML syntax for JSP directives, declarations, scriptlets, and expressions
  - Explain JSP interaction with HTML, servlets, and other JSP pages