Java Server Pages

Introduction

Agenda

1. Introduction JSP
2. Elements of JSP

Objectives

At the end of this module, you will be able to:

- Highlight benefits of JSP request model
- Distinguish JSP architecture models
- Understand the structure of a jsp page
- Use JSP constructs such as expressions, scriptlets, and declarations
- Create simple jsp pages

Introduction to JSP

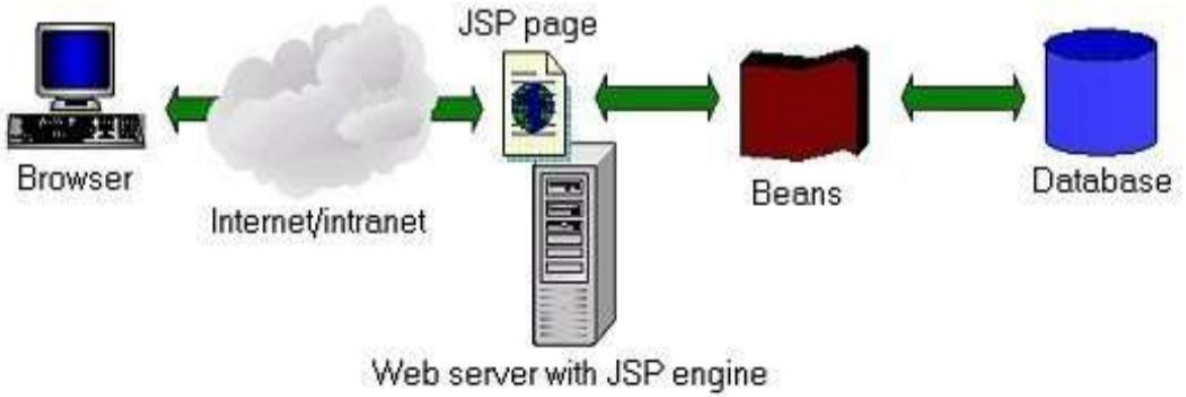
What is JSP?

- JSP stands for Java Server Pages, a technology invented by Sun to allow the easy creation of server side HTML pages.

- JSP lies in the presentation tier on the web server, with the main responsibility of generating HTML content that needs to be served to the browser. It also has the additional responsibility of pass on the requests to the backend through the JavaBeans, as and when required.

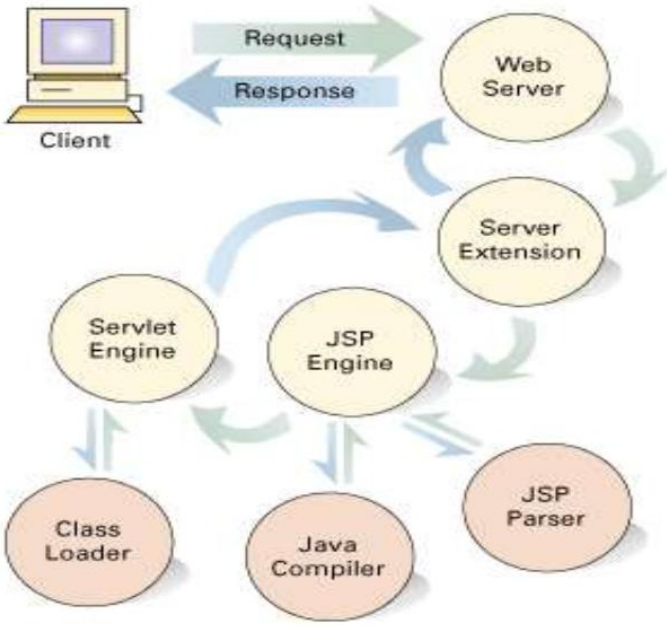
Why JSP?

1. High performance
2. Convenient to code and maintain as against servlets
3. Separates presentation from content
4. Powered by Java
   - Has access to all Java APIs
   - Has access to all J2EE APIsa
   - Inherent Platform independence



The Architecture: The flow of JSP request

1. The HTTP request comes to a web server.
2. The server extension receives the request (as it is not a request for static HTML but for a JSP) and passes it on to the JSP engine.
3. The JSP engine invokes the JSP parser to parse the JSP and check for any syntax errors.
4. On successful parsing, the JSP engine invokes the Java compiler to compile the JSP into an equivalent servlet class.
5. Once the class is generated, the control is passed on to a servlet engine.
6. The servlet engine loads the servlet into memory using its class loader.
7. The appropriate methods in the servlet are invoked and the response is routed back to the browser via the server extension and web server.



Elements of JSP

Scripting elements

- **Expressions:** Useful shorthand for printing out strings and contents of variables.
- **Scriptlets:** Lets you insert any valid Java code into the JSP.
- **Declarations:** Useful for declaring page wide variables and methods(Java) or functions.

Directives

- Affect the overall structure of the servlet class generated from this JSP
- format: `<%@ directive attribute="value" %>`
- There are two main types of directives: **page** & **include**

Actions

- Control the behavior of the servlet engine
- You can dynamically insert a file, reuse JavaBeans components
- Available actions include:
  - (1) jsp:include (2) jsp:forward (3) jsp:useBean

Scripting Elements: Expressions

A JSP *expression* is used to insert Java values directly into the output.

<%= Java Expression %>

- Example, the following shows the date/time that the page was requested: Current time: <%= new java.util.Date() %>
  - The Java expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at run-time (when the page is requested), and thus has full access to information about the request.
- Sample Example

```
<html>
<head><title>A first simple JSP</title></head>
<body>
<h3>
<%= “Welcome to JSP world!!!" %>
</h3>
</body>
</html>
```

Scripting Elements: Expressions - Pre-defined Variables

- To simplify the expressions, there are a number of predefined variables that you can use

- The most important ones are:
  - request
  - response
  - session
  - out

The Request Object Example

```
<html>
<head><title>Request Header Info</title></head>
<body bgcolor="white">
<ul>
<li>Request Method: <%=request.getMethod() %>
<li>Request URL    :<%=request.getRequestURI()%>
<li>Request Protocol :<%=request.getProtocol() %>
<li>Server Name :<%=request.getServerName()%>
<li>Server Port :<%=request.getServerPort()%>
<li>Remote Address: <%=request.getRemoreAddr() %>
<li>Browser :<%=request.getHeader("User-Agent") %>
</ul>
</body>
</html>
```

The out object Example

```
<html>
<head><title>The Out Object Example</title></head>
<body bgcolor="white">
<% out.print("<h3> JSP BufferInformation</h3>");
<%-- display the page buffer size --%>
Buffer:<%=out.getBufferSize() %> bytes <br/>
<%-- display the AutoFlush setting -- %>
AutoFlush: <%=out.isAutoFlush() %> <br/>
<%-- display the free page buffer space --%>
Remaining buffer : <%=out.getRemaining() %> bytes
</body>
</html>
```

The Response Object Example

```
<html>
<head><title>Response Object Info</title></head>
<body bgcolor="white">
<% response.setContentType("text/html"); %>
Buffer Size: <response.getBufferSize() %> bytes <br/>
Character Encoding :<%=response.getCharacterEncoding() %>
<br/>
Locale: <%=response.getLocale() %>
<%-- redirect the user to another.html --%>
<% response.sendRedirect("http://localhost:8080/another.html"); %>
</body>
</html>
```

The session object Example

```
<html>
<head><title>The Out Object Example</title></head>
<body bgcolor="white">
Session: <%=session.getId() %><hr/>
<% session.invalidate();%>
Session : <% if(session.getId() !=null)
    out.println(session.getId());
      else
    out.println("Session ended ") ; %>
</body>
</html>
```

Scripting Elements: Scriptlets

- Are defined as any block of valid Java code that resides between <% and %> tags
- Code that is defined within a scriptlet can access any variable and any beans that have been declared
- Scriptlets are like declarations in that they always use semicolons to end statements and expressions. They can have multiple expressions and statements as long as each is ended with a semicolon
- Scriptlets code goes into the service method of the JSP's compiled servlet which means it is executed only once when a request is actually serviced by the JSP

Scriptlets Example

```
<html>
<head><title>Scriptlet Example</title></head>
<body bgcolor="white">

<% java.util.Date now= new java.util.Date(); %>
Current date &amp;amp time :<%= now %>
<% if(now.getHourse()<12) { %> Good Morning!
<% } else if(now.getHours() <17 { %> Good Afternoon!
<% } else { %> Good Evening! <% } %>
</body>
</html>
```

Scripting Elements: Declaration

- Used to define methods or fields that get inserted into the main body of the servlet class
- It has the form: <%! Java Code %>
- The scope of a declaration is usually a JSP file, but if the JSP file includes other files with the include directive, the scope expands to cover the included files as well

Declaration Scripting Element

```
<html>
<head><title>Declaration sricpting element Example</title></head>
<body bgcolor="white">

<%! int counter=0; %>

Global counter :<%=++counter %>
</body>
</html>
```

Directives

- Affect the overall structure of the servlet class generated from this JSP
- format: <%@ directive attribute="value" %>
- There are two main types of directives:
  - page
  - include

Directives: Page Directives

- Defines attributes that apply to an entire JSP page

- Lets you do things like:
  - Import classes
  - Handle error messages
  - Define if the JSP is thread-safe
  - Define is session object is available
  - Set the page content type

Directives: The Include Directive

- Inserts the contents of another file in the main JSP file, where the directive is located
- Useful for including copyright information, scripting language files, or anything you might want to reuse in other applications
- The included file can be an HTML file, a JSP file, a text file, or a code file written in the Java programming language

Actions

- Control the behavior of the servlet engine
- You can dynamically insert a file, reuse JavaBeans components
- Available actions include:
  - jsp:include
  - jsp:forward
  - jsp:useBean

Actions: The jsp:include Action

- Lets you insert files into the page being generated
- The syntax looks like this:

```
<jsp:include page="relative URL" />
```

- Unlike the include directive, which inserts the file at the time the JSP page is translated into a servlet, this action inserts the file at the time the page is requested

The include element Example

```
<%@ page language="java" contentType="text/html" %>
<html>
<head><title>Include Action Demo</title></head>

<body bgcolor="white">

  <jsp:include page="nav.jsp" />

</body>
<html>
```

Actions: The jsp:forward Action

- Forwards a client request to an HTML file, JSP file, or servlet for processing

- Syntax

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

The forward element Example

```
<%
   double freeMem = Runtime.getRuntime().freeMemory();
   double totlMem = Runtime.getRuntime().totalMemory();
   double percent = freeMem/totlMem;

   if (percent < 0.5) {
%>
<jsp:forward page="ex2.jsp"/>

<% } else { %>

<jsp:forward page="ex3.jsp"/>

<% } %>
```

Quiz

1. A JSP page is transformed into _____.
2. JSP pages provide a means to create dynamic Web pages using HTML and the Java programming language. (TRUE/ FALSE)
3. With JSP, the coding is done in Java and only in Java.(TRUE/ FALSE)
4. Is JSP page extensible?(TRUE/FALSE)
5. JSP handles runtime errors using _____ attribute in page directive.
6. How do I use comments within a JSP page?
7. Identify the implicit objects in JSP

   a. *request* b. *response* c. *out*d. *session* e. *application*

Summary

In this module, you were able to:

- Describe the Software Component Assembly Model
- Explain Java's approach to developing software components
- Develop a simple Bean
- Distinguish JSP architecture vis-à-vis servlets
- Define and use the basic JSP Elements
- Create and use Java Beans

References

1. Hans Bergsten (2012). *Java Server Pages*, Retrieved on April 12, 2012, from, http://shop.oreilly.com/product/9781565927469.do
2. Stephanie Bodoff (2012). *JavaServer Pages Technology*. Retrieved on April 12, 2012, from, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro.html
3. coe.neu.edu (2012). *How JSP Works.* Retrieved on April 24 , 2012, from, http://www1.coe.neu.edu/~yozbek/isyg250/jsp.pdf
4. Marty Hall (1999). *Java Server Pages (JSP) 1.0.* Retrieved April 24, 2012, from, http://users.polytech.unice.fr/~buffa/cours/internet/POLYS/servlets/Servlet-Tutorial-JSP.html
5. Apl.jhu.edu (1999). *Java Server Pages(JSP) 1.0.*Retrieved April 24, 2012, from, http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-JSP.html
6. Paul Wheaton (2012). *Java - JSP*. Retrieved April 24, 2012, from, http://www.coderanch.com/t/281980/JSP/java/Hyperlinks

Thank You