Java Server Pages

CSE 486/586
Oct 05, 2004

References:
http://java.sun.com/products/jsp/
http://www.jsptut.com
Vijay Arthanari’s presentation on JSP
Introduction

- JSP technology allows creation of web content with both dynamic and static content

- It provides the dynamic capabilities of Java Servlets technology but are easier to write Most containers that support JSP will convert the JSP to a servlet class during deployment

- JSP supports a defined set of standard tags and user-defined custom tags
When to use Servlets

- Use servlets to implement services – servlet can perform whatever service it provides (templating, security, personalization, application control) then select and forward the request for display to a presentation component (JSP page).

- Use servlets as a web tier controller, which determines how to handle a request and chooses the next view to display.

- Avoid writing servlets that print mostly static text.
When to use JSP

- Use JSP pages for Data Presentation
- Not appropriate for creating content with highly variable structure or for controlling request routing.
- Use JSP pages to generate XML
- Use JSP pages to generate unstructured textual content such as ASCII text, fixed-width or delimited data, and even PostScript.
- Ideal for assembling textual data from multiple sources
What is a JSP page?

- A JSP page is a text document with two types of text:
  - Static template data expressed in any text-based format, such as HTML, SVG, WML and XML
  - JSP elements: Special markup for including other text or executing embedded logic which construct dynamic content.

- A JSP page services requests as a Servlet. The life cycle and many of the capabilities of JSP pages (in particular the dynamic aspects) are determined by Java Servlet technology.
JSP Elements

- There are basically three different forms of JSP elements:
  - Directives: Instructions that control the behavior of the JSP page compiler; evaluated at page compilation time.
  - Scripting elements: Blocks of Java code embedded in the JSP page between the delimiters <%= and %>.
  - Custom tags: Programmer-defined markup tags to generate dynamic content when the page is served.
JSP Elements

- Page directive: Defines attributes that apply to a JSP page. Example,
  - `<%@ page import="java.util.*, java.sql.*" %>`
- Include: Includes another JSP file within the current one
  - `<%@ include file="relativeURL" %>`
- Expression: Contains an expression that would get written onto the output page (HTML)
  - `<%= var1 + var2 %>`
- Scriptlet: Contains a code fragment
  - `<% Java code %>`
- Complete JSP syntax detailed at:
Example: HelloJSP (index.jsp)

```jsp
<%@ page import="java.sql.*" %>
<%! -- This is a comment in JSP. It will not be shown to the browser at all --%>
<%! -- This is a comment in HTML. This will be shown at the browser. So I might as well put my name here. :-%>

Written by: Mohit Vora (mhvora@buffalo.edu)
Teaching Assistant
Dept. of Computer Science & Engg.
Project: RMI & Webservices, CSE 4586, Fall 2004 -->

<html>
<head>
<title>A JSP Example</title>
<link rel="stylesheet" type="text/css" href="./format.css"/>
</head>
<body>
<h1>Using JSP to display records from the database</h1>
<% try { Class.forName("oracle.jdbc.driver.OracleDriver"); } catch (ClassNotFoundException e) { e.printStackTrace(); out.println("Database Driver cannot be loaded. Please check back later."); } %>
<int alternator = 0; try {
    Connection conn = DriverManager.getConnection(
            "jdbc:oracle:thin:mhvora/30865956@oraserve.cse.buffalo.edu:1521:csedb");
    Statement stmt = conn.createStatement();
```
Example: HelloJSP (index.jsp)

```jsp
ResultSet rs = stmt.executeQuery("select * from personal");

<table>
  <tr bgcolor="99ccff">
    <td>SSN</td>
    <td>First Name</td>
    <td>Last Name</td>
    <td>City</td>
    <td>Zip</td>
  </tr>
  <% while (rs.next()) {
    %>
  <tr bgcolor="<%= (alternator % 2 == 0) ? "#ffffff" : "99ccff" %>
    <td><%= rs.getInt("ssn") %></td>
    <td><%= rs.getString("firstname") %></td>
    <td><%= rs.getString("lastname") %></td>
    <td><%= rs.getString("city") %></td>
    <td><%= rs.getString("state") + ", " + rs.getString("zip") %></td>
  </tr><% alternator++ %>
  <% } %>
</table>
```

```jsp
catch (SQLException e) {
    e.printStackTrace();
    out.println("Error in SQL. Please check back later.");
}
```

```jsp
</html>
```
Generated Servlet (index_jsp.java)

- This file will get generated at
  $CATALINA_HOME/work/Catalina/localhost/<app-name>/org/apache/jsp/

- Looks something like:

```java
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.sql.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

private static java.util.Vector _jspx_dependants;

public java.util.List getDependants() {
    return _jspx_dependants;
}

public void _jspService(HttpServletRequest request, HttpServletResponse response)
    throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;
```

001/Hunter_et_al_jaxp.pdf
try {
    _jspxFactory = JspFactory.getDefaultFactory();
    response.setContentType("text/html");
    pageContext = _jspxFactory.getPageContext(this, request, response,
        null, true, 8192, true);
    _jspx_page_context = pageContext;
    application = pageContext.getServletContext();
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
    _jspx_out = out;

    out.write("\r\n");
    out.write("<html>");
    out.write("\r\n");
    out.write("<head>");
    out.write("\r\n");
    out.write("<title>A JSP Example</title>");
    out.write("\r\n");
    out.write("<link rel="stylesheet" type="text/css" href="./format.css"/>\r\n");
    out.write("\r\n");
    out.write("<body>");
    out.write("\r\n");
    out.write("<h1>Using JSP to display records from the database</h1>\r\n");
    out.write("\r\n");

    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
        out.println("Database Driver cannot be loaded. Please check back later.");
    }

    int alternator = 0;
try {
    Connection conn = DriverManager.getConnection("jdbc:oracle:thin:mhvora/30865956@oraserve.cse.buffalo.edu:1521:csedb");
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("select * from personal");
    out.write("<p>
              <font><table>
              	<tr bgcolor="99ccff">
              		<td><b>SSN</b></td>
              		<td><b>First Name</b></td>
              		<td><b>Last Name</b></td>
              		<td><b>City</b></td>
              		<td><b>Zip</b></td>
              	</tr>

    
    
    while (rs.next()) {
        out.write("<tr bgcolor="
        out.print( (alternator % 2 == 0)?"#ffffff":"99ccff" );
        out.write("<\tr><\td>\n}"
        out.print( rs.getInt("ssn") );
        out.write("<\td><\td>\n}
        out.print( rs.getString("firstname") );
        out.write("<\td><\td>\n}
        out.print( rs.getString("lastname") );
        out.write("<\td><\td>\n}
        out.print( rs.getString("city") );
        out.write("<\td>\n}"
}
```java
out.write("</td>\r\n");
    out.write("\r\t\t\t\t<td>\r\n");
    out.println(rs.getString("state") + ", " + rs.getString("zip"));
    out.write("\r\t\t\t\</tr>\r\n");
    alternator++;
    } conn.close();
    }
catch (SQLException e) { 
    e.printStackTrace();
    out.println("Error in SQL. Please check back later.");
    }
out.write("</table></font>\r\n");
out.write("</table></body>\r\n");
out.write("</html>"};
```
Working with Tomcat

- Web Applications and web services can be deployed in the following ways:
  - Copy .war file to $CATALINA_HOME/webapps
  - Copy application “build” directory to $CATALINA_HOME/webapps
  - Write XML configuration and copy to $CATALINA_HOME/conf/Catalina/localhost/

- Classes and libraries that are shared between applications go into $CATALINA_HOME/classes and $CATALINA_HOME/libs respectively
Web application / service

- Ideal directory structure of a web application / service
  - ApplicationRoot
    - /WEB-INF: Anything under here will not be world-viewable
      - /classes: Used to store class files that will be used in the application
      - /lib: Used to store jar files that contains classes that may be used
      - web.xml: Contains meta-information about the application
    - Web content (JSP, HTML, CSS, images): All files will world viewable. By default, index.jsp / index.html (if present) will be loaded when no file is specified
Tips

- To redeploy applications, undeploy using Manager Console and “ant deploy” application again.
- Undeploy other applications from Tomcat so that your server starts faster. Make sure you don’t take out the following:
  - admin
  - manager
  - saaj-related (for educational purposes)