**National University of Computer & Emerging Sciences, Karachi**

**Computer Science Department**

**Fall 2024, Lab Manual - 07**

|  |  |
| --- | --- |
| **Course Code: SL3001** | **Course: Software Development and construction** |
| **Instructor:** | **Yasir Arfat** |

**Lab # 07**

# Java Server Pages (JSP)

JavaServer Pages (JSP) is a server-side technology used for creating dynamic web pages. It enables the development of web applications by embedding Java code in HTML pages. JSP simplifies the development process by allowing developers to mix HTML with Java, providing a more flexible and maintainable way to build dynamic content.

Key Features of JSP:

- Separation of business logic from presentation layer

- Simplifies web development with minimal Java code

- Supports reusable Java components such as JavaBeans and custom tags

- Works seamlessly with databases, sessions, and HTTP requests

# Setting Up JSP

To run JSP applications, you'll need a Java-enabled web server like Apache Tomcat. Follow these steps to set up your environment:  
1. Download and install Apache Tomcat.  
2. Set up your project directory structure:  
 - WebContent/  
 - WEB-INF/  
 - web.xml  
 - index.jsp

# Basic JSP Tags

JSP (Java Server Pages) provides several types of tags that allow you to embed Java code into your HTML pages. Here are the main types:

1. **Page Directive Tag (<%@ page ... %>)**

The <%@ page %> directive in JSP is used to define various attributes and settings for the JSP page. It is a way to give the container specific instructions on how to handle the page, including its behavior, structure, and interaction with other resources.

**Example:**

<%@ page attribute="value" %>

You can use multiple attributes to control various aspects of the JSP page.

**Common Attributes of the Page Directive:**

**language**:

Specifies the scripting language used in the page. Typically set to "java", which is the default.

**Example**:

<%@ page language="java" %>

**contentType**:

Defines the MIME type and character encoding of the response. By default, it is text/html; charset=ISO-8859-1.

**Example**:

<%@ page contentType="text/html; charset=UTF-8" %>

**import**:

Allows importing Java classes, just like the import statement in Java.

**Example**:

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

**session**:

Controls whether the page can access the HTTP session object. The default is true. If you set it to false, the session object is not available.

**Example**:

<%@ page session="false" %>

**errorPage**:

Specifies the URL of the JSP page that will handle exceptions or errors thrown on the current page.

**Example**:

<%@ page errorPage="error.jsp" %>

**isErrorPage**:

Indicates whether the page itself is an error page. If set to true, the page can access exception objects using exception.

**Example**:

<%@ page isErrorPage="true" %>

**pageEncoding**:

Specifies the character encoding for the JSP page. It is essential for ensuring proper handling of text in various languages.

**Example**:

<%@ page pageEncoding="UTF-8" %>

1. **Expression Tag (<%= ... %>)**

The Expression Tag in JSP, written as <%= expression %>, is used to output the result of evaluating an expression directly to the client's web browser. It is a simple and effective way to display dynamic content, such as variables, method results, or calculations, on a webpage.

<%= expression %>

Here, the **expression** can be any valid Java expression, and the result of this expression will be inserted into the page at that position

**Example:**

<html>  
<body>  
<h1>Welcome to JSP</h1>  
<p>Today's date is: <%= new java.util.Date() %></p>  
</body>  
</html>

1. **Scriptlet Tag (<% ... %>)**

The Scriptlet Tag in JSP, written as <% Java code %>, is used to embed Java code within the JSP page. Scriptlets allow you to include any valid Java code, such as declarations, loops, conditionals, and method calls, directly in the JSP. The code inside a scriptlet is executed every time the page is requested by the client.

**How to Use the Scriptlet Tag:**

The syntax for a scriptlet tag is:

<% Java code %>

You can write any block of Java code inside the scriptlet tag. This code is processed on the server, and it can be used to manipulate data, perform operations, or control the flow of the page (e.g., conditionals or loops).

**Example:**

<html>  
<body>  
<h1>Welcome to JSP</h1>  
  
  
<%  
int num1 = 5;  
int num2 = 10;  
int sum = num1 + num2;  
%>  
  
<ul>  
 <%  
 for (int i = 1; i <= 5; i++) {  
 out.println("<li>Item " + i + "</li>");  
 }  
 %>  
</ul>  
  
  
</body>  
</html>

1. **Declaration Tag (<%! ... %>)**

The Declaration Tag in JSP, written as <%! ... %>, is used to declare variables and methods that can be used throughout the JSP page. Unlike scriptlets (<% ... %>), which are executed as part of the JSP's service() method, code inside a declaration tag is placed in the servlet class itself. This means declarations are initialized only once and are available across multiple requests.

**How to Use the Declaration Tag:**

The syntax for the declaration tag is:

<%! Java code (variable or method declarations) %>

You can declare fields (instance variables) or methods using the declaration tag, which can then be accessed and used in other parts of the JSP page.

**Example:**

<%!  
int count = 0; // Declaration of a variable  
public String greetUser(String name) { // Declaration of a method  
return "Welcome, " + name + "!";  
}  
%>  
  
<html>  
<body>  
<h1>Greeting Message</h1>  
<p><%= greetUser("Ahmed") %></p>  
</body>  
</html>

1. **Include Action Tag (<jsp:include ... />)**

IThe Include Action Tag (<jsp:include ... />) in JSP is used to include the content of another resource (like another JSP file, HTML file, or servlet) into the current JSP page at request processing time. This is useful for reusing common page components, such as headers, footers, or navigation menus, across multiple JSP pages without duplicating code.

**How to Use the <jsp:include> Tag:**

The syntax for the include action tag is:

<jsp:include page="relativeURL" />

**page:** Specifies the URL of the resource you want to include. The resource can be another JSP page, HTML file, or servlet, and the path is typically relative to the current JSP file.

**Example:**

<html xmlns:jsp="http://java.sun.com/JSP/Page">  
<body>  
<jsp:include page="header.jsp" />  
  
<h1>Main Content of the Page</h1>  
  
<jsp:include page="footer.jsp" />  
</body>  
</html>

1. **Forward Action Tag (<jsp:forward ... />)**

The Forward Action Tag (<jsp:forward ... />) in JSP is used to forward a request from one JSP page to another resource, such as another JSP page, servlet, or HTML page. This action transfers control to another resource without returning to the original page, meaning the client is unaware of the forward as it is handled entirely on the server side.

**How to Use the <jsp:forward> Tag:**

The syntax for the forward action tag is:

<jsp:forward page="relativeURL" />

page: Specifies the relative URL of the resource to which the request should be forwarded. This can be another JSP page, servlet, or HTML file.

**Example:**

<jsp:forward page="nextPage.jsp" xmlns:jsp="http://java.sun.com/JSP/Page"/>

In this example, the current request is forwarded to nextPage.jsp. Once forwarded, the execution of the current JSP page is halted, and the control is transferred to the specified page**.**

1. **Parameter Tag (<jsp:param ... />)**

Used The Parameter Tag (<jsp:param ... />) in JSP is used to pass parameters from one JSP page to another, or from a JSP page to a servlet. It is typically used in combination with other JSP actions like <jsp:forward>, <jsp:include>, or <jsp:plugin>. The <jsp:param> tag allows you to send key-value pairs (parameters) that can be accessed by the target page or resource.

**How to Use the <jsp:param> Tag:**

The syntax for the parameter tag is:

<jsp:param name="parameterName" value="parameterValue" />

name: The name of the parameter.

value: The value of the parameter.

The <jsp:param> tag is usually nested inside other action tags like <jsp:forward>, <jsp:include>, or <jsp:plugin> to pass parameters when forwarding or including content.

### Example:

### Example with <jsp:forward>:

<jsp:forward page="nextPage.jsp">

<jsp:param name="userName" value="Ahmed" />

<jsp:param name="userRole" value="admin" />

</jsp:forward>

In this example:

* The request is forwarded to nextPage.jsp, and the parameters userName with the value "Ahmed" and userRole with the value "admin" are passed to the forwarded page.

**Example with <jsp:include>:**

<jsp:include page="header.jsp">

<jsp:param name="title" value="Home Page" />

</jsp:include>

In this example:

* The header.jsp page is included, and the title parameter with the value "Home Page" is passed to it.

1. **UseBean Tag (<jsp:useBean ... />)**

The **UseBean Tag** (<jsp:useBean ... />) in JSP is used to create or locate a JavaBean in a JSP page. A JavaBean is a reusable software component that follows certain conventions, including having a no-argument constructor, providing getter and setter methods for properties, and being serializable. The <jsp:useBean> tag allows JSP pages to interact with these JavaBeans by creating a new instance or using an existing one.

**How to Use the <jsp:useBean> Tag:**

The syntax for the <jsp:useBean> tag is:

<jsp:useBean id="beanName" class="fullClassName" scope="beanScope" />

* **id**: The name by which the bean will be identified in the JSP page.
* **class**: The full class name of the JavaBean (including the package).
* **scope**: Specifies the lifecycle or scope of the bean (optional). The possible values are:
  + page (default): The bean is available only in the current JSP page.
  + request: The bean is available throughout the current request.
  + session: The bean is available for the entire session.
  + application: The bean is available to all JSP pages and servlets within the application.

**Example:**

<jsp:useBean id="user" class="com.example.User" scope="session" xmlns:jsp="http://java.sun.com/JSP/Page"/>

In this example:

* A JavaBean of the class com.example.User is either created or fetched from the session scope if it already exists.
* The bean is identified by the id="user" and can be accessed using this identifier within the JSP page.

**Basic Form handling and Java Servlet with Annotation**

Here's a simple example of form handling using **Servlets with annotations**. The example demonstrates how to submit form data (like username and password) to a Servlet, which then processes the data and responds back.

**1. JSP Form Page (form.jsp):**

This JSP page contains a basic HTML form that submits data (via POST) to a Servlet.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  
<html>  
<head>  
 <title>Form Handling with Servlet</title>  
</head>  
<body>  
  
<h2>Login Form</h2>  
  
<!-- Form that sends data to the Servlet -->  
<form action="login" method="POST">  
 <label for="username">Username:</label>  
 <input type="text" id="username" name="username" required><br><br>  
  
 <label for="password">Password:</label>  
 <input type="password" id="password" name="password" required><br><br>  
  
 <input type="submit" value="Login">  
</form>  
  
</body>  
</html>

**2. Servlet (LoginServlet.java):**

This is the Servlet that processes the form data submitted by the user. We'll use **Servlet annotations** to avoid needing to configure the web.xml manually.

package com.example;  
  
import java.io.IOException;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
// Using @WebServlet annotation to define the URL pattern  
@WebServlet(name = " LoginServlet", urlPatterns = {"/task5"})  
public class LoginServlet extends HttpServlet {  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
  
 // Retrieving form data sent by the user  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
  
 // Basic validation logic (this can be more complex in real applications)  
 if ("admin".equals(username) && "password".equals(password)) {  
 // If the credentials are correct, forward to a success page  
 request.setAttribute("message", "Welcome, " + username + "!");  
 request.getRequestDispatcher("success.jsp").forward(request, response);  
 } else {  
 // If the credentials are incorrect, forward back to the form with an error message  
 request.setAttribute("errorMessage", "Invalid username or password!");  
 request.getRequestDispatcher("form.jsp").forward(request, response);  
 }  
 }  
}

**3. Success Page (success.jsp):**

This is a simple JSP page that displays a success message after successful login.

<%@ **page** language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  
<html>  
<head>  
 <title>Login Successful</title>  
</head>  
<body>  
  
<h2>Login Successful</h2>  
  
<p><%= request.getAttribute("message") %></p>  
  
</body>  
</html>