

Java server page thorough with examples

Thought for 21s >

Let's build JSP from zero → pretty strong with lots of small examples.

1. What is JSP?

JSP (JavaServer Pages) = HTML pages + Java code on server-side.

- File extension: .jsp
- Runs inside a servlet container (Tomcat).
- Request comes → JSP is converted to a Servlet → compiled → executed → HTML sent to browser.

You use JSP mainly for **view layer (UI)** in MVC:

- **Servlet** = controller (logic, DB calls, set data)
 - **JSP** = view (display data using JSTL, EL)
 - **Model** = Java classes / DTO / Beans
-

2. JSP Life Cycle (Interview favorite)

1. **Translation:** JSP → Java Servlet source (e.g. index_jsp.java)
2. **Compilation:** Servlet source → .class
3. **Loading & Instantiation:** Class loaded, object created
4. **Initialization:** jspInit() method
5. **Request processing:** _jspService(request, response) (called for every request)
6. **Destroy:** jspDestroy() when removed/unloaded

You normally don't call these methods; container does it.

3. Basic JSP Example

hello.jsp

jsp

 Copy code

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

```

<head>
    <title>Hello JSP</title>
</head>
<body>
    <h1>Hello from JSP</h1>
    <p>Current time: <%= new java.util.Date() %></p>
</body>
</html>

```

- `<%@ ... %>` → **directive**
- `<%= ... %>` → **expression**

4. JSP Directives

Syntax:

jsp

 Copy code

```
<%@ directiveName attributeName="value" ... %>
```

Main directives:

1. page
2. include
3. taglib

4.1 page Directive

Controls page-level settings.

Common attributes:

- import – import Java classes
- contentType – MIME type
- errorPage – error handling
- isErrorPage – mark page as error page
- session – enable/disable session

Example:

jsp

 Copy code

```
<%@ page import="java.util.List, java.time.LocalDate" %>
<%@ page contentType="text/html;charset=UTF-8" %>
```

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

Error Page Example

errorPage.jsp :

jsp

 Copy code

```
<%@ page isErrorPage="true" %>
<html>
<body>
<h2>Oops! Something went wrong.</h2>
<p>Error: <%= exception.getMessage() %></p>
</body>
</html>
```

main.jsp :

jsp

 Copy code

```
<%@ page errorPage="errorPage.jsp" %>
<%
    int x = 10 / 0; // will cause exception
%>
```

4.2 include Directive (Static include)

Includes content at **translation time** (like copy-paste).

jsp

 Copy code

```
<%@ include file="header.jsp" %>
<body>
    Page Content
</body>
<%@ include file="footer.jsp" %>
```

- If header.jsp changes → need recompile/reload to reflect (depends on container).
 - Good for common layout (header, footer, menu).
-

4.3 taglib Directive

Used to use libraries like **JSTL**.

jsp

 Copy code

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Then:

jsp

 Copy code

```
<c:out value="${message}" />
```

We'll use this in JSTL section.

5. Scripting Elements

OLD STYLE (still used in interviews, but in real apps we prefer JSTL + EL).

Types:

- 1. Scriptlet:** <% ... %>
- 2. Expression:** <%= ... %>
- 3. Declaration:** <%! ... %>

5.1 Scriptlet

Java code inside JSP.

jsp

 Copy code

```
<%  
    int a = 10;  
    int b = 20;  
    int sum = a + b;  
%>  
<p>Sum is: <%= sum %></p>
```

Avoid heavy logic here, keep logic in Servlet / Service.

5.2 Expression

Output value directly.

jsp

 Copy code

```
<p>Welcome, <%= request.getParameter("username") %></p>
<p>Time: <%= new java.util.Date() %></p>
```

Equivalent to:

java

 Copy code

```
out.print( value );
```

5.3 Declaration

Declares methods/variables as **instance members** of generated Servlet.

jsp

 Copy code

```
<%
    int counter = 0;
    public int getNextCounter() {
        return ++counter;
    }
%>

<p>Counter: <%= getNextCounter() %></p>
```

Used rarely in modern code.

6. Implicit Objects in JSP

Available automatically (no need to create):

1. `request` → `HttpServletRequest`
2. `response` → `HttpServletResponse`
3. `session` → `HttpSession`
4. `application` → `ServletContext`
5. `out` → `JspWriter`
6. `page` → `this` (current JSP instance)
7. `config` → `ServletConfig`
8. `pageContext` → `PageContext` (gives access to all scopes)

9. exception → Throwable (only on error pages)

Example using some implicit objects

jsp

 Copy code

```
<%@ page import="java.util.Enumeration" %>
<html>
<body>
<h2>Request Parameters</h2>
<ul>
<%
    Enumeration<String> names = request.getParameterNames();
    while (names.hasMoreElements()) {
        String name = names.nextElement();
        String value = request.getParameter(name);
    }
    <li><%= name %> = <%= value %></li>
<%
<%
    }
%>
</ul>

<p>Session ID: <%= session.getId() %></p>
<p>App Context Path: <%= application.getContextPath() %></p>
</body>
</html>
```

7. JSP Actions

Use XML-like tags: `<jsp:....>`.

Common ones:

1. `<jsp:include>`
2. `<jsp:forward>`
3. `<jsp:param>`
4. `<jsp:useBean>` , `<jsp:setProperty>` , `<jsp:getProperty>`

7.1 `<jsp:include>` – Dynamic include

Included at **request time** (can change per request).

[jsp](#)[Copy code](#)

```
<jsp:include page="header.jsp" />  
Content here  
<jsp:include page="footer.jsp" />
```

You can also pass parameters:

[jsp](#)[Copy code](#)

```
<jsp:include page="header.jsp">  
    <jsp:param name="title" value="Home Page" />  
</jsp:include>
```

header.jsp :

[jsp](#)[Copy code](#)

```
<h1><%= request.getParameter("title") %></h1>
```

7.2 <jsp:forward>

Forward request to another resource.

[jsp](#)[Copy code](#)

```
<%  
    if (request.getParameter("user") == null) {  
%>  
    <jsp:forward page="login.jsp" />  
    <%  
    }  
%>
```

With params:

[jsp](#)[Copy code](#)

```
<jsp:forward page="showDetails.jsp">  
    <jsp:param name="id" value="101"/>  
</jsp:forward>
```

7.3 JavaBean with <jsp:useBean>

Assume a Bean:

java

 Copy code

```
// src/com/demo/User.java
package com.demo;

public class User {
    private String name;
    private int age;

    public User() {}

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

JSP:

jsp

 Copy code

```
<%@ page import="com.demo.User" %>

<jsp:useBean id="user" class="com.demo.User" scope="request" />
<jsp:setProperty name="user" property="name" value="Parth" />
<jsp:setProperty name="user" property="age" value="22" />

<p>Name: <jsp:getProperty name="user" property="name" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
```

Or fill from form:

jsp

 Copy code

```
<jsp:setProperty name="user" property="*" />
```

This maps request parameters to bean properties with same name.

8. Expression Language (EL)

EL allows you to write `${...}` instead of Java code.

Used with attributes in scopes:

- page
- request
- session
- application

Priority: pageScope → requestScope → sessionScope → applicationScope

8.1 Basic Syntax

jsp

 Copy code

```
<p>Hello, ${param.username}</p>
<p>Session ID: ${session.id}</p>
<p>Request attribute 'msg': ${requestScope.msg}</p>
<p>Cookie user: ${cookie.user.value}</p>
<p>Context path: ${pageContext.request.contextPath}</p>
```

Common EL objects:

- param , paramValues
- header , headerValues
- cookie
- initParam
- pageScope , requestScope , sessionScope , applicationScope

8.2 Operators in EL

- Arithmetic: + - * / %
- Logical: and or not
- Relational: == != < > <= >=
- Empty: empty

Example:

jsp

 Copy code

```
<c:if test="${empty param.username}">
    <p>Username is required</p>
</c:if>
```

9. JSTL (JavaServer Pages Standard Tag Library)

Very important for real-world JSP.

Add JSTL dependency (if using Maven) or JAR in `WEB-INF/lib`.

Taglib for core:

jsp

 Copy code

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Common tags:

- `<c:out>` – safe output
- `<c:if>` – if condition
- `<c:choose>`, `<c:when>`, `<c:otherwise>`
- `<c:forEach>` – loop
- `<c:set>` – set variables
- `<c:remove>` – remove attributes

9.1 Example: if & choose

jsp

 Copy code

```
<c:set var="marks" value="75" />

<c:if test="${marks >= 60}">
    <p>First division</p>
</c:if>

<c:choose>
    <c:when test="${marks >= 75}">
        <p>Distinction</p>
    </c:when>
    <c:when test="${marks >= 60}">
        <p>First division</p>
    </c:when>
    <c:otherwise>
        <p>Needs improvement</p>
    </c:otherwise>
</c:choose>
```

9.2 Example: forEach (loop through List)

Assume Servlet:

java

□ Copy code

```
List<String> products = Arrays.asList("Laptop", "Phone", "Tablet");
request.setAttribute("products", products);
request.getRequestDispatcher("products.jsp").forward(request, response);
```

products.jsp :

jsp

□ Copy code

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<h2>Product List</h2>
<ul>
    <c:forEach var="p" items="${products}">
        <li>${p}</li>
    </c:forEach>
</ul>
```

For List of Beans:

java

□ Copy code

```
public class Product {
    private int id;
    private String name;
    private double price;
    // getters & setters
}
```

Servlet:

java

□ Copy code

```
List<Product> list = ...; // from DB
request.setAttribute("productList", list);
request.getRequestDispatcher("showProducts.jsp").forward(request, response);
```

showProducts.jsp :

jsp

□ Copy code

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>



| ID                                                                                                                                                                                     | Name                                                                                                      | Price                  |                            |                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|------------------------|----------------------------|------------------------------|
| <c:foreach &gt;<="" items="\${productList}" td="" var="p"><td><td><p.id}< p=""></p.id}<></td><td><p.name}< p=""></p.name}<></td><td><p.price}< p=""></p.price}<></td></td></c:foreach> | <td><p.id}< p=""></p.id}<></td> <td><p.name}< p=""></p.name}<></td> <td><p.price}< p=""></p.price}<></td> | <p.id}< p=""></p.id}<> | <p.name}< p=""></p.name}<> | <p.price}< p=""></p.price}<> |


```

10. Small MVC Flow Example (Servlet + JSP)

10.1 login.jsp (view – form)

jsp

 Copy code

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Login</title></head>
<body>
<h2>Login</h2>

<form action="login" method="post">
    Username: <input type="text" name="username" /><br/>
    Password: <input type="password" name="password" /><br/>
    <input type="submit" value="Login" />
</form>

<%-- Show error if present --%>
<% String err = (String) request.getAttribute("error"); %>
<% if (err != null) { %>
    <p style="color:red;"><%= err %></p>
<% } %>
```

```
</body>
</html>
```

(You can also use EL: \${error} .)

10.2 LoginServlet (controller)

java

 Copy code

```
// package com.demo;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.IOException;

public class LoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
            throws ServletException, IOException {

        String user = request.getParameter("username");
        String pass = request.getParameter("password");

        if ("admin".equals(user) && "123".equals(pass)) {
            HttpSession session = request.getSession();
            session.setAttribute("user", user);
            response.sendRedirect("home.jsp");
        } else {
            request.setAttribute("error", "Invalid credentials");
            RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
            rd.forward(request, response);
        }
    }
}
```

10.3 home.jsp (after login, using EL)

jsp

 Copy code

```
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>
```

```

<body>
<c:if test="${empty sessionScope.user}">
    <c:redirect url="login.jsp"/>
</c:if>

<h2>Welcome, ${sessionScope.user}!</h2>
<a href="logout">Logout</a>
</body>
</html>

```

10.4 LogoutServlet (just to connect your earlier work)

java

 Copy code

```

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.IOException;

public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }
        response.sendRedirect("login.jsp");
    }
}

```

11. Quick Comparison: Static HTML vs JSP vs Servlet

- **HTML:** Fixed content, no server logic.
- **Servlet:** All Java code; we write HTML inside Java → messy for views.
- **JSP:** HTML + tags + EL → better for UI.
- **Modern pattern:** Servlet / Controller sets data (`request.setAttribute`) → JSP displays using JSTL & EL.

If you want, next I can give you:

- Pure JSP interview question list (definitions + small code)
- Or notes style summary (1–2 lines per concept) that you can revise before exams.



Make a form addition using this

Sure

Here is a **complete Form Addition example using JSP** based on what you just learned.

I'll show you 2 ways:

1. Addition in SAME JSP page
2. Addition using Servlet + JSP (MVC way – professional)

1. Simple JSP Form Addition (Same Page)

add.jsp

jsp

Copy code

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
    <title>Addition using JSP</title>
</head>
<body>

<h2>Add Two Numbers (JSP)</h2>

<form method="post">
    Enter First Number:
    <input type="number" name="num1" required /><br/><br/>

    Enter Second Number:
    <input type="number" name="num2" required /><br/><br/>

    <input type="submit" value="Add" />
```

```

</form>

<%
    String n1 = request.getParameter("num1");
    String n2 = request.getParameter("num2");

    if (n1 != null && n2 != null) {
        int a = Integer.parseInt(n1);
        int b = Integer.parseInt(n2);
        int sum = a + b;
    }
%>
<h3>Result: <%= a %> + <%= b %> = <%= sum %></h3>
<%
%>

</body>
</html>

```

📌 Flow:

- User enters numbers
- Submits form
- JSP reads parameters
- Calculates and displays sum

2. Professional Way: Servlet + JSP (MVC Structure)

Step 1: addition.jsp (View)

jsp

 Copy code

```

<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <title>Addition Form</title>
</head>
<body>

<h2>Add Two Numbers</h2>

<form action="add" method="post">

```

```
Number 1: <input type="number" name="num1" required /><br/><br/>
Number 2: <input type="number" name="num2" required /><br/><br/>
<input type="submit" value="Add" />
</form>

<h3>Result: ${result}</h3>

</body>
</html>
```

Step 2: AddServlet.java (Controller)

java

 Copy code

```
package com.demo;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.IOException;

@WebServlet("/add")
public class AddServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response
        throws ServletException, IOException {

        int a = Integer.parseInt(request.getParameter("num1"));
        int b = Integer.parseInt(request.getParameter("num2"));

        int sum = a + b;

        request.setAttribute("result", sum);
        RequestDispatcher rd = request.getRequestDispatcher("addition.jsp");
        rd.forward(request, response);
    }
}
```

Output Example

User enters:

```
10  
20
```

 Copy code

Result:

```
makefile
```

 Copy code

Result: 30

EL-only Clean Version (Best Practice)

No scriptlets 

```
jsp
```

 Copy code

```
<form method="post">  
    Number 1: <input type="number" name="n1" />  
    Number 2: <input type="number" name="n2" />  
    <input type="submit" value="Add" />  
</form>  
  
<c:if test="${not empty param.n1}">  
    <c:set var="sum" value="${param.n1 + param.n2}" />  
    Result: ${sum}  
</c:if>
```

Interview Point

Q: How to perform addition in JSP?

- ✓ Using request parameters
- ✓ Convert to integers
- ✓ Perform arithmetic
- ✓ Display using Expression or EL

If you want, I can also show:

- ✅ Addition with session storing result
- ✅ Addition with database saving
- ✅ Addition with Ajax (without page reload)

Just tell me what style you want 😊



I need this is different jsp pages

Got it 👍

You want **Addition form using DIFFERENT JSP pages.**

Here are the two correct ways:

✓ METHOD 1: Two JSP Pages Only (NO Servlet)

1 input.jsp (Form Page)

jsp

Copy code

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
    <title>Addition Form</title>
</head>
<body>

<h2>Add Two Numbers</h2>

<form action="result.jsp" method="post">
    Enter First Number:
    <input type="number" name="num1" required /><br/><br/>

    Enter Second Number:
    <input type="number" name="num2" required /><br/><br/>

    <input type="submit" value="Add" />
</form>
```

```
</body>
</html>
```

2 result.jsp (Result Page)

jsp

 Copy code

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
    <title>Addition Result</title>
</head>
<body>

<h2>Result Page</h2>

<%
    String n1 = request.getParameter("num1");
    String n2 = request.getParameter("num2");

    int a = Integer.parseInt(n1);
    int b = Integer.parseInt(n2);

    int sum = a + b;
%>

<p><b><%= a %></b> + <b><%= b %></b> = <b><%= sum %></b></p>

<a href="input.jsp">Go Back</a>

</body>
</html>
```

✓ Flow:

css

 Copy code

```
input.jsp ---> result.jsp
(form)           (calculation)
```

✓ METHOD 2: Professional MVC Way (2 JSP + Servlet)

1 addForm.jsp

jsp

[Copy code](#)

```
<html>
<body>

<h2>Addition Form</h2>

<form action="add" method="post">
    Number 1: <input type="number" name="num1" required /><br/><br/>
    Number 2: <input type="number" name="num2" required /><br/><br/>
    <input type="submit" value="Add" />
</form>

</body>
</html>
```

2 AddServlet.java

java

[Copy code](#)

```
package com.demo;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.IOException;

@WebServlet("/add")
public class AddServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        int a = Integer.parseInt(request.getParameter("num1"));
        int b = Integer.parseInt(request.getParameter("num2"));

        int sum = a + b;

        request.setAttribute("result", sum);
    }
}
```

```
request.setAttribute("a", a);
request.setAttribute("b", b);

RequestDispatcher rd = request.getRequestDispatcher("result.jsp");
rd.forward(request, response);

}

}
```

3 result.jsp

jsp

 Copy code

```
<html>
<body>

<h2>Result</h2>

<p>
    ${a} + ${b} = <b>${result}</b>
</p>

<a href="addForm.jsp">Add Again</a>

</body>
</html>
```

✓ Flow:

 Copy code

addForm.jsp → AddServlet → result.jsp

✓ Which should you use?

Method	Use Case
JSP → JSP	Practice / basics

Method	Use Case
JSP + Servlet	Real project / Interview
<p><input checked="" type="checkbox"/> For CDAC / interviews, METHOD 2 is preferred.</p>	

Want Next Level?

I can also give you:

- Addition with **Session storage**
- Addition with **AJAX (no reload)**
- Scientific calculator JSP project
- Multiple operations (+ - × ÷)

Just tell me what you want 



[Explain method 1](#)

Great  Let's deeply explain **METHOD 1** (Addition using ONLY Two JSP Pages).

This is the simplest and most important concept for JSP beginners.

METHOD 1: Two JSP Pages (input.jsp → result.jsp)

No servlet is used.

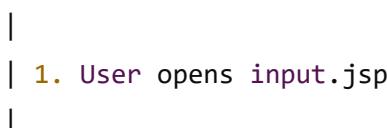
One JSP takes input, the other JSP processes and shows output.

Overall Flow (Important for understanding)

pgsql

 Copy code

Browser



```
input.jsp (form)
|
| 2. User enters numbers & clicks Add
|
POST request with num1 & num2
|
result.jsp (calculation)
|
| 3. Server calculates and returns HTML
|
Browser displays result
```

1 input.jsp – Form Page (Only UI)

jsp

 Copy code

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
    <title>Addition Form</title>
</head>
<body>

<h2>Add Two Numbers</h2>

<form action="result.jsp" method="post">
    Enter First Number:
    <input type="number" name="num1" required /><br/><br/>

    Enter Second Number:
    <input type="number" name="num2" required /><br/><br/>

    <input type="submit" value="Add" />
</form>

</body>
</html>
```

🔍 What happens here?

1. <form action="result.jsp">

- Tells browser: send data to `result.jsp` when button is clicked.
- 2.** `method="post"`
- Data goes securely in request body.
- 3.** Input fields:

html

Copy code

```
<input type="number" name="num1">
<input type="number" name="num2">
```

These names create request parameters:

ini

Copy code

```
num1=10
num2=20
```

So after clicking **Add**, browser sends:

bash

Copy code

```
POST /result.jsp
num1=10&num2=20
```

2 result.jsp – Processing + Output Page

jsp

Copy code

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <title>Addition Result</title>
</head>
<body>

<h2>Result Page</h2>

<%
    String n1 = request.getParameter("num1");
    String n2 = request.getParameter("num2");

    int a = Integer.parseInt(n1);
```

```

int b = Integer.parseInt(n2);

int sum = a + b;

%>

<p><b><%= a %></b> + <b><%= b %></b> = <b><%= sum %></b></p>

<a href="input.jsp">Go Back</a>

</body>
</html>

```

Line-by-Line Explanation

1. Getting data from input.jsp

jsp

 Copy code

```

String n1 = request.getParameter("num1");
String n2 = request.getParameter("num2");

```

- `request` is an implicit JSP object.
- `getParameter()` reads form values sent by browser.

So if user typed:

ini

 Copy code

```

num1 = 10
num2 = 20

```

Then:

ini

 Copy code

```

n1 = "10"
n2 = "20"

```

2. Converting String → int

jsp

 Copy code

```
int a = Integer.parseInt(n1);
int b = Integer.parseInt(n2);
```

Because:

- All form data is received as **String**.
 - We convert it to integer to perform arithmetic.
-

3. Performing Addition

jsp

 Copy code

```
int sum = a + b;
```

This is plain Java logic.

4. Displaying Result

jsp

 Copy code

```
<p><b><%= a %></b> + <b><%= b %></b> = <b><%= sum %></b></p>
```

Here:

- `<%= ... %>` prints output to browser.
- So the result becomes HTML, e.g:

 Copy code

10 + 20 = 30

Summary of Method 1

Step	What Happens
input.jsp	Displays form
User clicks Add	Browser sends data
result.jsp	Reads num1 & num2

Step	What Happens
result.jsp	Converts & adds
result.jsp	Shows result

💡 Interview Points

You can say this in exam:

- In JSP-only addition, one page collects input using HTML form.
- The form submits data to another JSP page using request parameters.
- The second JSP retrieves data using `request.getParameter()`, converts it to integer, performs arithmetic, and displays the result using JSP expression.

✓ Optional: Safe Version (Avoid error)

If user does not enter value:

jsp

 Copy code

```

<%
String n1 = request.getParameter("num1");
String n2 = request.getParameter("num2");

if(n1 != null && n2 != null) {
    int a = Integer.parseInt(n1);
    int b = Integer.parseInt(n2);
    int sum = a + b;
}
Result: <%= sum %>
<%
}
```

If you want, I can now show:

- Same example using EL instead of scriptlet
- Flow diagram for exam presentation

- ◆ Addition with subtraction/multiplication/division in 2 JSP pages

Just tell me 



All jsp actions

Here is a **complete, clear explanation of ALL JSP Action Tags** with syntax and examples



(These are very important for exams & interviews)



What are JSP Actions?

JSP Actions are **built-in XML-style tags** used to control the behavior of the JSP engine at request time.

They perform tasks like:

- Including resources
- Forwarding requests
- Working with JavaBeans
- Passing parameters

They start with:

makefile

 Copy code

<jsp: ... >



LIST OF ALL JSP ACTION TAGS

JSP Action	Purpose
<jsp:include>	Include another resource dynamically
<jsp:forward>	Forward request to another resource

JSP Action	Purpose
<jsp:param>	Pass parameters
<jsp:useBean>	Create / locate Java Bean
<jsp:setProperty>	Set bean property
<jsp:getProperty>	Get bean property
<jsp:plugin>	Embed applet/bean (obsolete)
<jsp:fallback>	Shown if plugin fails
<jsp:element>	Create dynamic XML element
<jsp:attribute>	Define attribute dynamically
<jsp:body>	Define body dynamically
<jsp:text>	Output plain text

Now let's cover each one with examples 👇

1 <jsp:include> – Dynamic Include

Includes another page at **request time**.

jsp

 Copy code

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

With parameter:

jsp

 Copy code

```
<jsp:include page="header.jsp">
  <jsp:param name="title" value="Home Page"/>
</jsp:include>
```

header.jsp:

jsp

 Copy code

```
<h2><%= request.getParameter("title") %></h2>
```

Difference from directive include:

- directive: static (compile time)
 - jsp:include: dynamic (runtime)
-

2 <jsp:forward> – Forward Request

Transfers control to another page.

jsp

 Copy code

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

With parameters:

jsp

 Copy code

```
<jsp:forward page="result.jsp">
    <jsp:param name="id" value="101"/>
</jsp:forward>
```

Stops current page execution.

3 <jsp:param> – Pass Parameters

Used inside include/forward.

jsp

 Copy code

```
<jsp:include page="show.jsp">
    <jsp:param name="user" value="Parth"/>
</jsp:include>
```

Access:

jsp

 Copy code

```
<%= request.getParameter("user") %>
```

4 <jsp:useBean> – Create/Locate Java Bean

Creates object of a class.

jsp

 Copy code

```
<jsp:useBean id="user" class="com.demo.User" scope="request"/>
```

Equivalent to:

java

 Copy code

```
User user = new User();
```

Scopes:

- page
- request
- session
- application

5 <jsp:setProperty> – Set Bean Property

Set value into bean.

jsp

 Copy code

```
<jsp:setProperty name="user" property="name" value="Parth"/>
```

From form automatically:

jsp

 Copy code

```
<jsp:setProperty name="user" property="*"/>
```

6 <jsp:getProperty> – Get Bean Property

Display bean value.

jsp

 Copy code

```
Name: <jsp:getProperty name="user" property="name"/>
```

✓ Complete Bean Example

Bean class:

java

 Copy code

```
package com.demo;

public class User {
    private String name;
    private int age;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

JSP:

jsp

 Copy code

```
<jsp:useBean id="u" class="com.demo.User" scope="request"/>
<jsp:setProperty name="u" property="name" value="Parth"/>
<jsp:setProperty name="u" property="age" value="22"/>

<p>
Name: <jsp:getProperty name="u" property="name"/><br>
Age: <jsp:getProperty name="u" property="age"/>
</p>
```

7 <jsp:plugin> – Embed Applet (Deprecated)

Used for Java Applets (rare now).

jsp

 Copy code

```
<jsp:plugin type="applet" code="MyApplet.class" width="300" height="200">
<jsp:fallback>
    Applet not supported

```

```
</jsp:fallback>  
</jsp:plugin>
```

8 <jsp:fallback>

Used inside plugin if browser does not support applet.

jsp

 [Copy code](#)

```
<jsp:fallback>  
    Your browser does not support this plugin.  
</jsp:fallback>
```

9 <jsp:element> – Create Dynamic XML Tag

jsp

 [Copy code](#)

```
<jsp:element name="note">  
    Hello World  
</jsp:element>
```

Produces:

xml

 [Copy code](#)

```
<note>Hello World</note>
```

10 <jsp:attribute> – Dynamic Attribute

jsp

 [Copy code](#)

```
<jsp:element name="person">  
    <jsp:attribute name="id">101</jsp:attribute>  
    Name: Parth  
</jsp:element>
```

Output:

xml

 [Copy code](#)

```
<person id="101">Name: Parth</person>
```

1 1 <jsp:body> – Dynamic Body

Defines content dynamically.

jsp

 Copy code

```
<jsp:element name="msg">
  <jsp:body>
    Welcome to JSP
  </jsp:body>
</jsp:element>
```

1 2 <jsp:text> – Output Plain Text

Outputs text exactly without parsing.

jsp

 Copy code

```
<jsp:text>
  This will not be interpreted as JSP code
</jsp:text>
```

Quick Interview Summary

You can say:

JSP Action elements are XML tags used to control request processing in JSP. They perform tasks like including resources, forwarding requests, and working with JavaBeans. Examples include `jsp:include`, `jsp:forward`, `jsp:useBean`, `jsp:setProperty`, and `jsp:getProperty`.

Most Important for Exams



Focus more on:

`jsp:include`