Java Server Page (JSP)

JSP - Introduction

✓**Java Server Pages (JSP)** is a server-side technology used to create static and dynamic web applications. The static content is expressed by text-based format files such as HTML, XML whereas JSP elements are used to construct dynamic content.

✓ A JSP page consists of **HTML tags and JSP tags**. The JSP pages are easier to maintain than Servlet because we can **separate designing and development**.

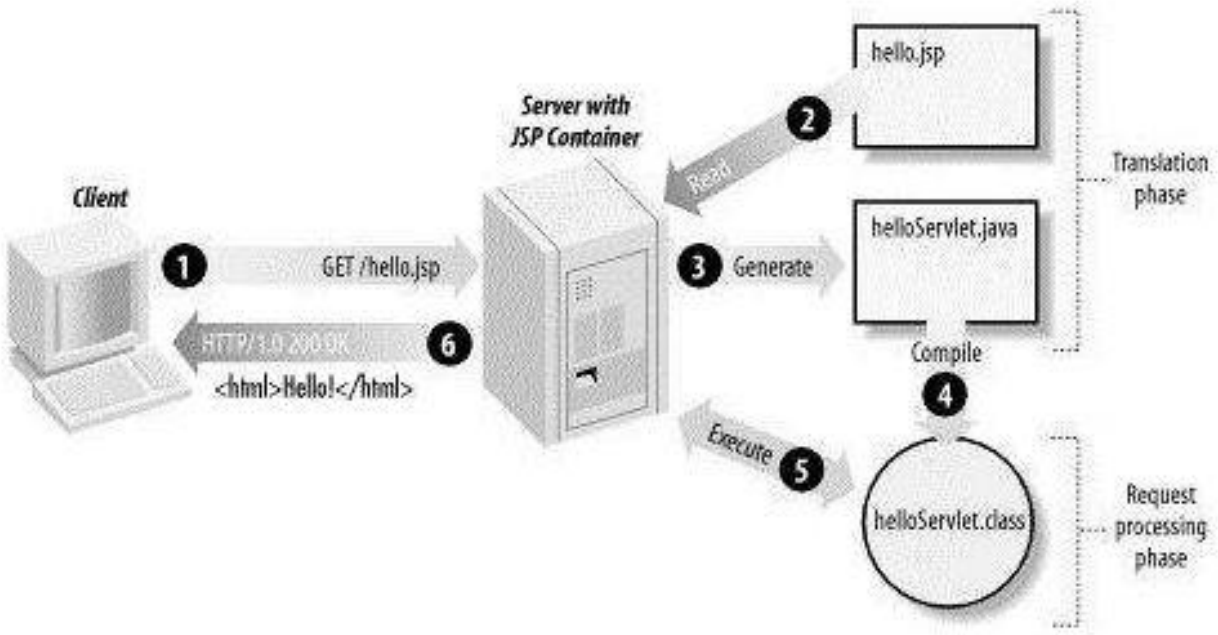
✓JSP is a convenient way of writing servlets. It enables you **to insert Java code into HTML pages** through simple JSP tags. JSP source files are represented by **.jsp extension**.

✓The web server needs a JSP engine, i.e, a container to process JSP pages.

The JSP container is responsible for intercepting requests for JSP pages.

✓ JSP Container knows how to understand the special elements that are part of JSPs.

JSP - Processing



- browser sends an HTTP request to the web server.
- The JSP engine **loads the JSP page** from disk and **converts it into a servlet content**.
- The JSP engine compiles the servlet into an executable class and **forwards the original request to a servlet engine**.
- web server called the servlet engine **loads the Servlet class** and executes it.
- the servlet **produces an output** in HTML format.
- web server **forwards the HTTP response** to your browser in terms of static HTML content.
- the web browser **handles the dynamically-generated HTML page** inside the HTTP response

JSP - Example

```
<%@ page import = "java.util.*" %>
<html>
  <head>
    <title>Display Current Date & Time</title>
  </head>
  <body>
    <center>
      <h1>Display Current Date & Time</h1>
    </center>
    <%
      Date date = new Date();
      %>
          <p>  <% date.toString() %> </p>
  </body>
</html>
```

JSP Scripting Elements

✓JSP scripting provides an easy and efficient way **to insert Java programming language in JSP pages**. Here, Java is a default language.

✓SP provides **three types of scripting elements** to embed Java code in JSP page. Each scripting element has its own purpose.

1.Scriptlet tag

2.Declaration tag

3.Expression tag

JSP – Scriplet Tag

✓JSP scriptlet tag allows you **to insert programming logic inside JSP page**. Hence, you can put valid Java code within scriptlet tags. Each Java statement must be followed by a semicolon.

Syntax: - *<% Java code %>*

✓Whatever the code written inside scriptlet tag <% %> **is compiled as Java code**. This code can be accessible anywhere inside the JSP page.

✓Any number of Java statements and local variables can be inserted in scriptlet tag. Although, **scriptlet doesn't allow method declaration within it**.

```
<html>
<head>
<title>Welcome to JSP </title>
</head>
<body>
<h2><center>Welcome to JSP
<%
java.util.Date date = new java.util.Date();
out.print("<br>Last Accessed Time: "+date);
%>
```

```
</center></h2>
</body>
</html>
```

JSP – Declaration Tag

✓JSP declaration tag is used **to declare variables and methods in JSP page**. Unlike scriptlet tag, JSP container placed the content of declaration tag **outside the jspService() method**. So, variables and methods declared in declaration tag are static or instance.

✓Syntax: **<%! variable and method declaration %>**

```
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<%!
String name=“VIT";
int count=0;
%>
<%out.print(“Institute name:"+name);
out.print("<br>Number of time visited:"+ ++count);
%>
</body>
</html>
```

JSP – Expression Tag

JSP expression tag allows you **to place the result of Java expression** in a convenient way. It can be seen as an alternative of out.print() method.
For Example, if we want to print **any statement or result directly** then we can use the below syntax:

<%=result%>

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h1><center>
<%="Welcome to JSP tutorial" %>
</center></h1>
</body>
</html>
```

JSP – Read Form Data

```
<html>
  <head>
    <title>Using GET Method to Read Form Data</title>
  </head>

  <body>
    <h1>Using GET Method to Read Form Data</h1>
    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>

  </body>
</html>
```

JSP - Directives

✓ In JSP, the role of directive is to provide directions to the JSP container regarding the compilation of JSP page.

✓ Directives convey information from JSP page to container that give special instruction at translation time.

The **three types** of directive provided by JSP is as follows: -

1. **Page Directive**
2. **Include Directive**
3. **Taglib Directive**

Include Directive

✓As the name implies, include directive is **used to include the content of other files** such as HTML, JSP, text into the current JSP page.

✓These files are included during translation phase.

✓Syntax: **<@ include file="file-name">**

JSP – Include Directive

```
<h1 align="center">
Welcome to JSP
</h1>
```

```
<h5 align="center">
This is JSP include directive
</h5>
```

```
<html>
<head>
<title>Welcome</title>
</head>
<body>
<%@ include file="header.html"%>
<font size="5">
<p align="center">Learn JSP tutorials</p></font>
<%@ include file="footer.html"%>
</body>
</html>
```

JSP – Page Directive

JSP page directive provides various attributes with unique properties. These attributes can be applied to an entire JSP page.

The syntax of Page directive is as follows: -

<@page attribute="value">

Attributes of Page Directive

- ✓**contentType -** use contentType attribute to define MIME type and character set of the response message.

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

  <@page buffer="value">

- ✓**extends -** The extends attribute inherits the superclass to serve its properties in current class.

  <@page extends="package.class">

- ✓**import -** This attribute is used to import packages in JSP page.

  <@page import="package-name“>

JSP – Taglib Directives

Taglib directive is used to define tag libraries in JSP page. It enables user **to use custom tags within JSP file**. A JSP page can contains more than one taglib directives.

<@ taglib uri="uri" prefix="value">

Here, *uri* represents the *path of tag library* description and *prefix* represent the *name of custom tag*.

```
</html>
<head>
<title> TagLib Example</title>
</head>
<body>
<h2 align="center">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:out value="Welcome to JSP Taglib Directive"/>
</h2>
</body>
</html>
```

JSTL – Java Standard Tag Library

✓ Java Standard tag library (JSTL) is a **collection of various type of JSP tags**. Each tag control the behavior of JSP page and **perform specific task** such as database access, conditional execution etc.

✓ STL delivers the functionality of programming methodology without using Java code. Thus, it provides **ease to develop and maintain web applications**.

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

Types of JSTL tag

In JSP, JSTL tags are divided into following parts: -

1.JSTL Core - core tags are most frequently used tags that provides variable support, manages URL and control the flow of JSP page

2.JSTL Formatting - to specify the type of format that represents date and time.

3.JSTL SQL - to access and manipulate various operations of database. It is responsible to interact the JSP page with any type of Database such as MySQL, Oracle, SQLite, etc.

4.JSTL XML - for creating and manipulating xml documents.

5.JSTL function - to the various string methods in Java.

JSP – Implicit Objects

✓In JSP, implicit objects are pre-defined objects, created by the web container. These objects are created during the translation phase from JSP to Servlet.
✓JSP allows you to call these objects directly without initializing them.
✓Implicit objects are required to be declared in scriptlet tags only.

✓JSP provides 9 implicit objects that can be used in JSP pages. Each object represents some unique identity.

out	javax.servlet.jsp.JspWriter
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
config	javax.servlet.ServletConfig
application	javax.servlet.ServletContext
session	javax.servlet.http.HttpSession
exception	javax.servlet.http.HttpException
Page	java.lang.object
PageContext	javax.servlet.jsp.PageContext

JSP – Implicit Objects

```
<html>
<head>
<title> Implicit Object Example</title>
</head>
<body>
<%
String s="VTOP";
out.println("Learn JSP");
out.println("<br>Website:"+s);
int i1=out.getBufferSize();
out.println("<br>Buffer Size:"+i1);
int i2=out.getRemaining();
out.println("<br>Remaining Size:"+i2+"</h2>");
%>
</body>
</html>
```

JSP – Action Tags

JSP action tags are used to perform various actions during the execution of JSP page. Here, each action performs some specific task.

These tasks include:

- ✓ ***Forwarding the current page request to another page.***
- ✓ ***Including external page to JSP page.***
- ✓ ***Creating a JavaBean instance.***

JSP provides additional functionality to JSP pages through these action tags. Unlike directive tags, action tag conveys the information to servlet container when request is being processed.

Syntax:

<jsp:action_name attribute="value">

JSP – Action Tags

List of JSP Action Tags

These are the various action tags provided by JSP: -

- **jsp:forward** – This tag is used to forward the request of current JSP page to any another JSP, Servlet or HTML page.
- **jsp:include** – This tag includes the external resource to JSP page.
- **jsp:useBean** – This tag deals with the object of JavaBean.
- **jsp:setProperty** – This tag sets the property of JavaBean object.
- **jsp:getProperty** – This tag gets the property associated with JavaBean object.
- **jsp:text** – This tag is used to write template text in JSP page.
- **jsp:elements** – This tag is to define XML elements dynamically.
- **jsp:attribute** – This tag defines the attribute of dynamically generate XML elements.
- **jsp:body** – This tag defines the body of dynamically generate XML elements.
- **jsp:plugin** – This tag is used to integrate Java components with JSPpage.

Example

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

Error: Password must be more than 4 digits <br>
 <jsp:include page="index.jsp"></jsp:include>
```

JSP – Database Access

```
<%@ page import="java.sql.*" %>
<html>
<head><title>Database Connection Example</title></head>
<body>
<%
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;

  try {
    // Load JDBC Driver
    Class.forName("com.mysql.jdbc.Driver");

    // Establish Connection
    con = DriverManager.getConnection(
      "jdbc:mysql://localhost:3306/DatabaseName", "username", "password");
```

JSP – Database Access

```
// Create Statement
    stmt = con.createStatement();

    // Execute Query
    rs = stmt.executeQuery("SELECT * FROM users");

    // Process Result Set
    while(rs.next()) {
      out.println("Name: " + rs.getString("name") + "<br>");
    }
  } catch (ClassNotFoundException e) {
    out.println("Driver not found: " + e.getMessage());
  } catch (SQLException e) {
    out.println("SQL Error: " + e.getMessage());
  }
}
%>
</body>
</html>
```

Database Access in JSP using Tags

SQL Tags

JSTL SQL tag library can be used to interact with database. **Tab library provides tag to perform several operations on database.**

· URI for format tag library is http://java.sun.com/jsp/jstl/sql

· To use SQL tag library on JSP page we need to import its tag library using

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

1. <sql:setDataSource>

In any application where we need to interact with database, we need to first **create a datasource to provide all the required information about database** so our application is aware about the database to connect.

Following attributes are supported by this tag-

· **driver-** this attribute tells the driver class to be loaded and registered to interact with database.

· **url-** URL for the database connection

· **user-** username of Database

· **password-** password of Database

· **var-** variable to store datasource information

· **scope-** scope in which variable will be saved. Possible values are Page, request, session and application. Default value is Page.

SCOPE

Database Access using JSP tags

Syntax

```
<sql:setDataSource var="libraryDataSource" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/LIBRARY"
    user="user_id"  password="password“
/>
```

2. <sql:query>

This tag is used to fire the select query and saves the result in scoped variable.
Following attributes are supported by this tag-

- **sql-** to provide the select query. Alternatively we can add the select statement in body of tag
- **datasource-** database connection . This attribute should not be used with <sql:query> tag is used within <sql:transaction> tag
- **var-** variable to store result of query
- **scope-** scope in which variable will be saved. Possible values are Page ,request, session and application . Default value is Page.

```
<sql:query var="booksQuery" datasource=”libraryDatasource”
            select * from books;
</sql:query>
```


SCOPE

Database Access in JSP

3. <sql:update>

This tag is used to fire the insert, delete or update select query and saves the result(affected rows) in scoped variable.

```
<sql:update var="booksQuery" datasource=”libraryDatasource”
            delete  from books;
</sql:update>
```

4. <sql:param>

This tag is used with <sql:query> and <sql:update> to supply a value for a value placeholder

Syntax

<sql:param value="scoped_variable" />

5.<sql:transaction>

This tag is used to provide transactions functionality and used with <sql:query> and <sql:update> . All the statements added in this tag becomes the part of same transactions (which means either all changes are performed or none will be performed)

Syntax of this tag is

<sql:transaction dataSource="libraryDatasource"> “>

Database Access in JSP

```
<html>
 <head>
  <title> sql:insert tag example </title>
 </head>
 <%@ taglib prefix="c"   uri="http://java.sun.com/jsp/jstl/core" %>
 <%@ taglib prefix="sql"  uri="http://java.sun.com/jsp/jstl/sql" %>
 <body>
  <sql:setDataSource var="libraryDataSource" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost:3306/LIBRARY"
     user="root"  password="password"/>
  <sql:update dataSource="${libraryDataSource}" var="affectedRows">
     insert into books values
     ('ISBN1234', 'JSP Tutorial', '124.3','Joe Bloggs');
  </sql:update>
  Number of Rows Inserted are :: <c:out value="${affectedRows }"/>
 </body>
</html>
```

Database Access in JSP

```
<html>
 <body>
  <sql:setDataSource var="libraryDataSource" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/LIBRARY"
    user="root"  password="password"/>
  <sql:query dataSource="${libraryDataSource}" var="records">
    select * from books;
  </sql:query>
  <table border="1" >
   <tr>
    <th>ISBN Number</th>
    <th>Name of Book</th>
    <th>Price in $ </th>
    <th>Author</th>
   </tr>
   <c:forEach var="row" items="${records.rows}">
   <tr>
    <td><c:out value="${row.isbn}"/></td>
    <td><c:out value="${row.name}"/></td>
    <td><c:out value="${row.price}"/></td>
    <td><c:out value="${row.author}"/></td>
   </tr>
```

Thank you