

Unit-5 JSP Java Server Page

5.1Relation of Applets and Servlets with JSP

5.2 JSP Scripting Elements

5.3 JSP Expressions

5.4 Difference between JSP and Servlets

5.5 JSP Declarations

5.6 Simple JSP program to fetch database records

❖ Introduction

- **JSP** technology is used to create web application, used to dynamically generate HTML/XML just like Servlet technology.
- It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc.
- 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.
- It provides some additional features such as Expression Language, Custom Tag etc.
- Goal of JSP is the simplified creation and management of dynamic web pages.
- The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags.

Advantage of JSP over Servlet

There are many advantages of JSP over servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

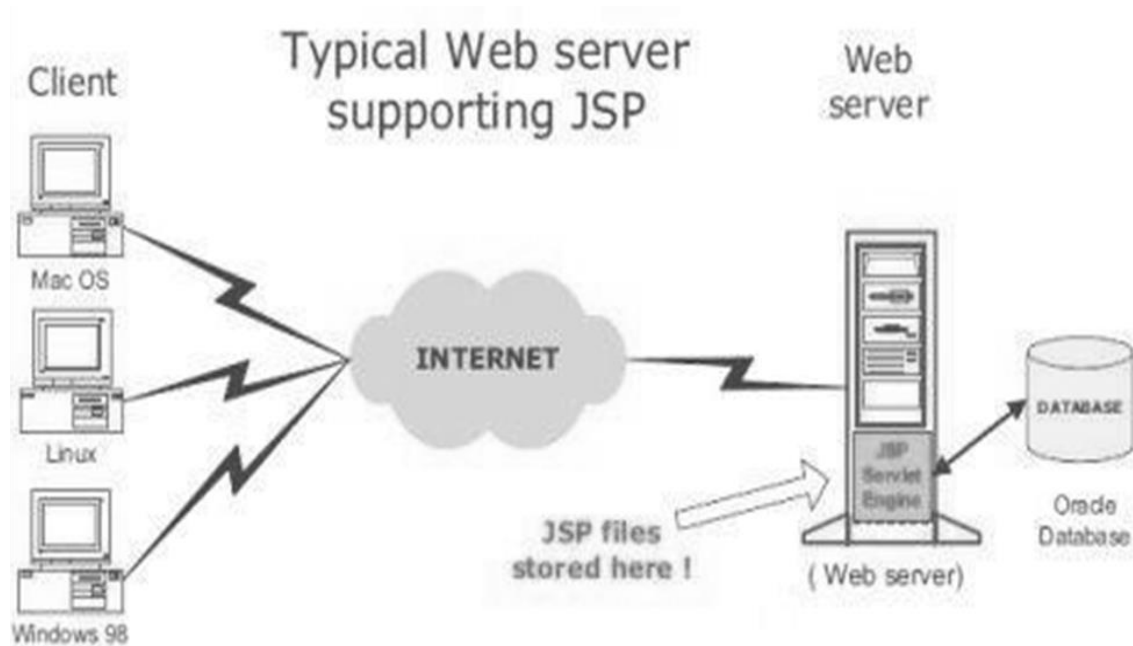
4) Less code than Servlet

In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

❖ JSP Processing

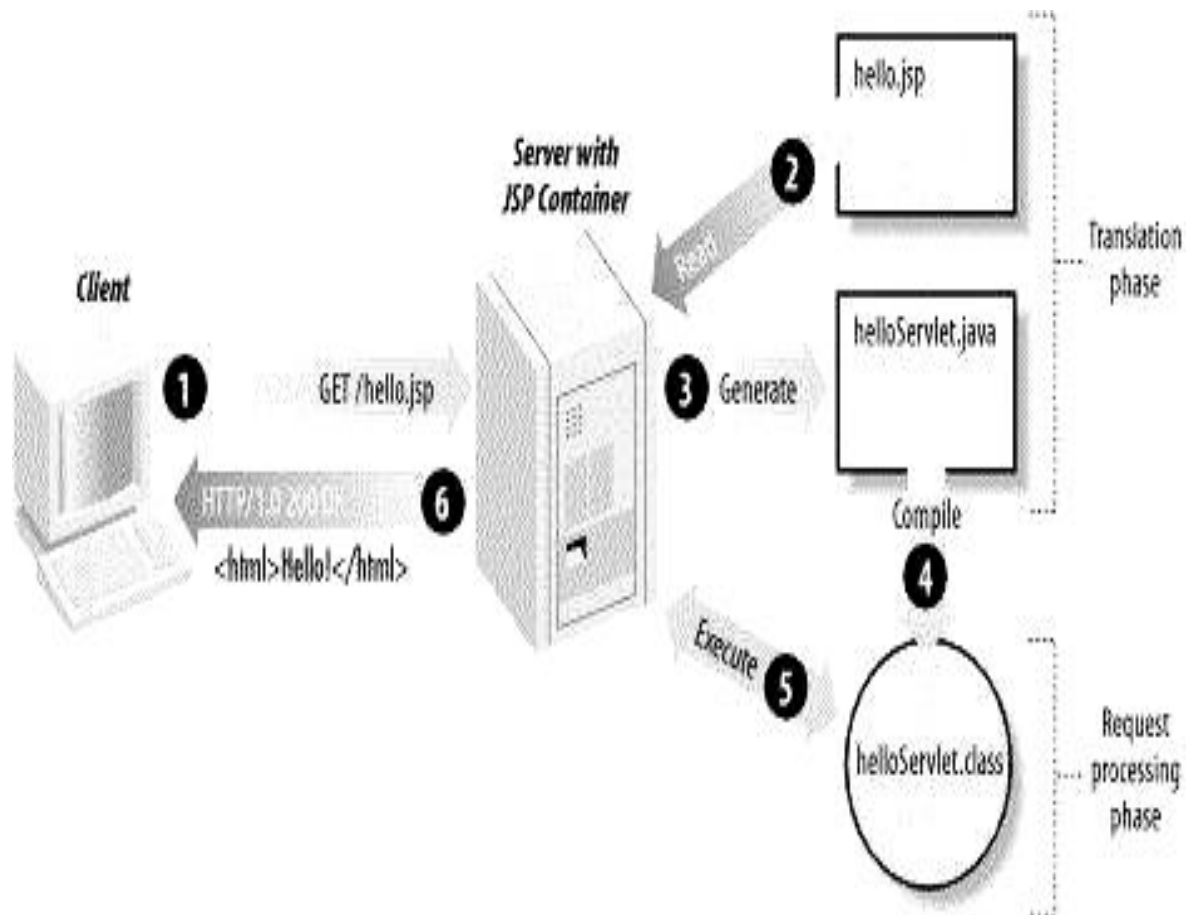
- A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

- Following diagram shows the position of JSP container and JSP files in a Web Application.



Steps to how web server creates the web page using JSP:

- 1) As with a normal page, your browser sends an HTTP request to the web server.
- 2) The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**.
- 3) The JSP engine loads the JSP page from disk and converts it into servlet content. This conversion is very simple in which all template text is converted to `println ()` statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- 4) The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- 5) A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- 6) The web server forwards the HTTP response to your browser in terms of static HTML content.



Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

- Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents.
- This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster. So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming. Except for the translation phase, a JSP page is handled exactly like a regular servlet

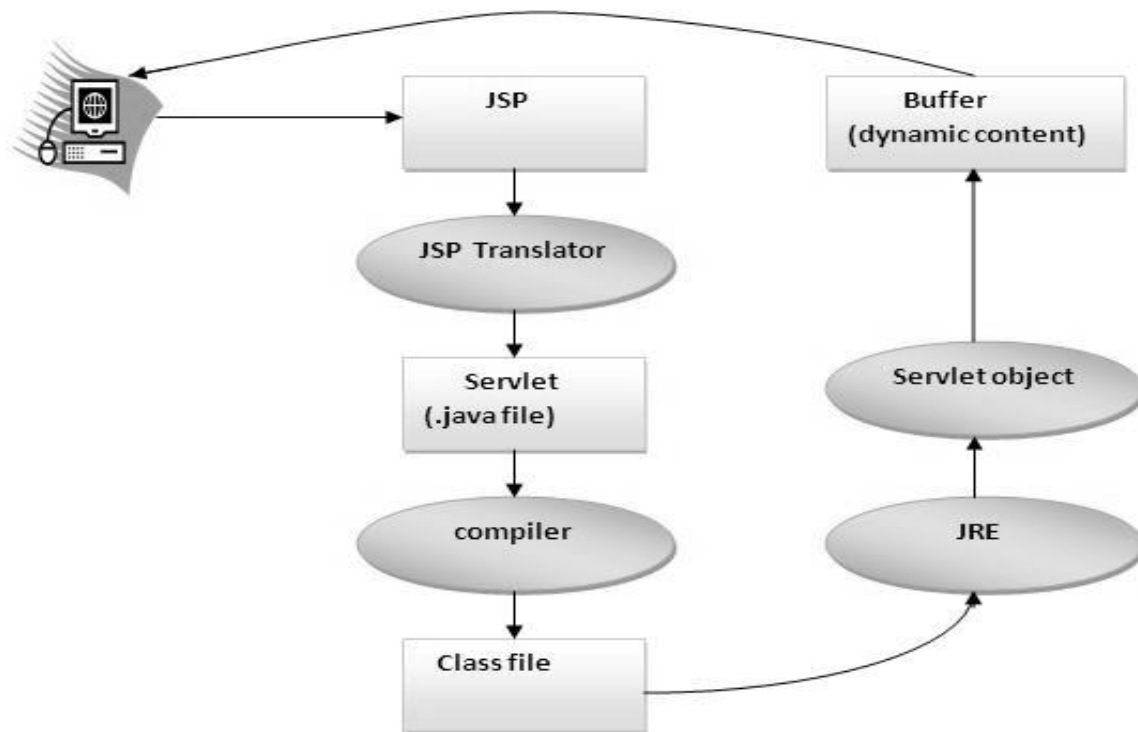
❖ Life cycle of a JSP Page

- JSP page looks like a HTML page but is a servlet.
- It is not executed directly.
- A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

The JSP page follows these phases:

1. **Translation** of JSP Page
 2. **Compilation** of JSP Page
 3. **Class loading** (class file is loaded by the classloader)
 4. **Instantiation** (Object of the Generated Servlet is created).
 5. **Initialization** (jspInit() method is invoked by the container).
 6. **Request processing** (_jspService() method is invoked by the container).
 7. **Destroy** (jspDestroy() method is invoked by the container).
1. **Translation:**
 - JSP container checks the JSP page code and convert it to generate the servlet source code. You can find generated servlet class files at <tomcat>/WEB-INF/org/apache/jsp.
 - After JSP page is translated into servlet all the request for that JSP page are served by its related servlet class.
 2. **Compilation of JSP Page**
 - In this Java source code for the related servlet is compiled by the JSP container. The container converts source code into byte code.
 3. **Class loading :** in this servlet class file is loaded by the class loader.
 4. **Instantiation :** Object of the Generated Servlet is created.
 5. **Initialization:**
 - In this instantiated object is initialized. For this **jspInit()** method is invoked by the container.
 - jspInit() method declared in JSP page and it is implemented by JSP container. This method called once in JSP life cycle to initialize it with config params configured in deployment descriptor.
 - We can override this method using JSP declaration scripting element to initialize any resource that we want to use in our JSP page.
 6. **Request processing :**
 - In this stage of JSP life cycle only those objects of a JSP equivalent servlet that initialized successfully are used by the web container to handle client request.
 - **_jspService ()** method is invoked by the container for each client request by passing request and response object.

- This method cannot be override. All the jsp code goes inside this method and it is overridden by default.
- This method defined in `HttpJspPage` interface.



7. Destroy :

- If a servlet container decides to destroy a servlet instance of a JSP page , **`jspDestroy()`** method is invoked by the container.
- It clean up the environment .It is called only once in JSP life cycle.

Note: `jspInit()`, `_jspService()` and `jspDestroy()` are the life cycle methods of JSP.

As depicted in the above diagram,

- JSP page is translated into servlet by the help of JSP translator.
- The JSP translator is a part of web server that is responsible to translate the JSP page into servlet.
- After that Servlet page is compiled by the compiler and gets converted into the class file.
- Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.

❖ **Relation of Applet and Servlet with JSP**

- There are three primary ways to create client interfaces to java programs that are delivered over the internet : **applet , servlets and Java Server Page(JSP)**
- Applets run within a client browser, while servlets and JSPs run on the web server machine.

Applets

- Applets are small programs that are downloaded into a Java enabled web browser.
- The browser will execute applet on clients machine .The downloaded applet has very limited access to client's machine's file system and network capabilities.
- Applet greatly enhances the user interface available through browser.
- But applet applets need to be downloaded over the internet. Browser must download the whole application to execute it. The more functionality the applet provides the longer it will take download.
- Applets are best suited for high functionality applications where download times are not factor or don't require special security access.

Servlet

- It is a program that runs with a web server. A servlet is executed in response to an HTTP request from a client browser.
- The servlet executes and then returns an HTML page back to browser.
- Once it started it remains in memory and can handle multiple HTTP requests. It provides dynamic content.
- They deliver HTML pages to their clients, the user interface provided through a servlet is limited.

JSP

- JSP are text document that describe how a server should handle specific requests.
- JSP is run by JSP server, which interprets the JSP and performs the actions the page describes.
- JSP server compiles the JSP into a servlet to enhance performance. The server would the periodically check the JSP for changes. If JSP changes the server will recompile it into a servlet.
- JSP is extension to servlet.
- JSP looks very similar to static HTML pages , expect they contain special tags are used to identify and define areas that contain java functionality.JSP are easy to develop and maintain because they do not need to be compiled.
- However because they need to be interpreted or compiled by the server , response time for initial accesses may be slower than a servlet.

❖ **Difference between JSP and servlet**

No.	JSP	Servlet
1	JSP is a file which consists of HTML tag and scripting code.	Servlet is a java class.
2	Files are saved using .jsp extension	Files are saved using .java extension
3	It is server side scripting language which is used to develop dynamic web pages.	Servlet is special program of JAVA that is already compiled and used to create dynamic web contents.
4	JSP run slower as compared to Servlet because we need to compile the code written in JSP to convert it in servlet code.	Servlet runs faster as compared to JSP because servlet is already compiled.
5	Coding is easy in JSP as compared to Servlet.	Coding is difficult in servlet as compared to JSP.
6	JPS are generally not preferred when there is much processing of data required	Servlets are preferred when there is large amount data processing required.
7	JSP are supported to HTTP protocol only.	Servlets are supported to HTTP,FTP and SMTP protocols.
8	We do not have to recompile and reloading JSP after modification, it will automatically reflected.	For every modification done in servlet program, we need to recompile and reload the application.
9	In JSP we can create custom tag which can directly call Java Beans.	In servlet there is no facility of custom tag.

❖ JSP elements

- It consists of various elements or tags
 - **Scripting tag**
 - **Implicit object**
 - **Directives**

[1] Scripting tag (elements)

- In JSP, java code can be written inside the JSP page using the scriptlet tag
- The scripting elements provides the ability to insert java code inside the JSP
There are mainly three types of scripting elements:
 1. **scriptlet** tag (`<% %>`)
 2. **expression** tag (`<%= %>`)
 3. **declaration** tag (`<%! %>`)
 4. **comment** tag (`<%-- --%>`)
- Declarations in JSP are processed at HTTP translation time and are available to other declarations, expressions and scriptlet in compiled JSP file.
- Expressions in JSP are also evaluated at HTTP translation time. The value of each expression is converted to a string and inserted in place in compiled JSP file.

1) Scriptlet tag

- It is used to execute java source code in JSP.
- A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.
- Scriptlet evaluated at request processing time, using the values of any declarations and expressions that are made to them.
- It is used to do complex code rather than simple expression scriptlet tag is used.

Syntax of Scriptlet: `<% code fragment %>`

- You can write scriptlet tag in XML as follows:

`<jsp:scriptlet>`

Code fragment

`</jsp: scriptlet>`

- Any text, HTML tags, or JSP elements you write must be outside the scriptlet.
Following is the simple and first example for JSP:

Example-1 Write a program to print Hello world in JSP

```
<html>
<head><title>JSP Hello World</title>
</head>
<body>
<%
    out.println("Hello World<br>");
%>
</body>
</html>
```

Output: Hello World

Example-2: Write a JSP program to print today's date.

```
<%@page import="java.util.*"%>

<html>
<head><title>JSP Hello World</title>
</head>
<body>
<%
    Date dt = new Date();
    out.println("Today's date :"+dt);
%>
</body>
</html>
```

Output: Today's date : Wed Apr 13 21:51:47 IST 2016

2) Declaration tag

- A declaration declares one or more variables or methods that you can use in Java code later in the JSP file.
- You must declare the variable or method before you use it in the JSP file.
- Declarations do not generate any output, they are normally used in conjunction with JSP expressions or scriptlet tags.

Syntax of JSP Declarations:

```
<%! Declaration; declaration; %>
```

- You can write declaration tag in XML file as follows:

<jsp:declaration>

Code fragment

</jsp:declaration>

Following is the simple example for JSP Declarations:

```
<%! int i=0;      %>
<%! int a,b,c;    %>
<%! Circle a=new Circle(2.0); %>
```

Example-3

```
<html>
<head><title>JSP Hello World</title>
</head>
<body>
<%! int a=2;int b=3; %>
      value of a=<%=a%>          </br>
      value of b=<%=b%>          </br>
</body>
</html>
```

Output: value of a=2
 value of b=3

3) Expression tag

- A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.
- Because the value of an expression is converted to a String, you can use an expression within a line of text, whether or not it is tagged with HTML, in a JSP file.
- The expression element can contain any expression that is valid according to the Java Language Specification but **you cannot use a semicolon to end an expression.**
- Syntax of JSP Expression:

```
<%= expression %>
```

You can write XML equivalent of the above syntax as follows:

<jsp:expression>**Expression****</jsp:expression>**

Following is the simple example for JSP Expression:

Example-4 Write a JSP program which performs addition of two variables.

```
<html>
<head><title>JSP Hello World</title>
</head>
<body>
    <%! int a=2;int b=3; %>
    sum of a and b= <%= a+b %>
</body>
</html>
```

Output :

sum of a and b=5

Example-5 Write a JSP program to print today's date.

```
<html>
<head><title>A Comment Test</title></head>
<body>
<p>
    Today's date: <%= (new java.util.Date()).toLocaleString() %>
</p>
</body>
</html>
```

Output: Today's date: 8 Apr, 2015 10:11:18 PM

Example-6 Write JSP code to find maximum out of two values.

```
<html>
<head><title>JSP Hello World</title>
</head>
<body>

    <%! int a=23;int b=30; %>

    <% if (a>b){ %>
        <%=a%> is maximum
    <% } else { %>
        <%=b%> is maximum
    <% } %>
</body>
</html>
```

Output: 30 is maximum

Example -7 Write a program to add two values using method.

```
<html>
<head><title>Method Declaration</title>
</head>
<body>
    <%!
        int sum(int a,int b)
        {
            return (a+b);
        }
    %>

    <%= "sum="+sum(10,200)
    %>
</body>
</html>
```

Output: sum =210

4) JSP Comments:

- JSP comment marks text or statements that the JSP container should ignore.
- A JSP comment is useful when you want to hide or "comment out" part of your JSP page.

Syntax:

```
<%- - comment - -%>
```

Example -8 Following is the simple example for JSP Comments:

```
<html>
<head><title>A Comment Test</title></head>
<body>
<h2>A Test of Comments</h2>
<%-- This comment will not be visible in the page source --%>
</body>
</html>
```

[2] Directive tags

- JSP directives provide directions and instructions to the container, telling it how to handle certain aspects of JSP processing.
- A JSP directive affects the overall structure of the servlet class. It usually has the following form:

<%@ directive attribute="value" %>

Ex : **<%@ page import="java.util.*"; %>**

For XML

<jsp:directive.directiveType attribute=value />

Ex: **<jsp:directive.page import="java.util.*" />**

- Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.
- The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

There are three types of directive tag:

Sr. No.	Directive	Description
1	<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
2	<%@ include ... %>	Includes a file during the translation phase.
3	<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page

1) JSP page Directive:

- The **page** directive is used to provide instructions to the container that pertain to the current JSP page.
- You may code page directives anywhere in your JSP page. By convention, page directives are coded at the top of the JSP page.

Following is the basic syntax of page directive:

```
<% @ page attribute="value" %>
```

Following is the list of attributes associated with page directive:

Attribute	Purpose
Buffer	Specifies a buffering model for the output stream.(8kb default)
autoFlush	Controls the behavior of the servlet output buffer.(true false)
contentType	Defines the character encoding scheme.
errorPage	Defines the URL of another JSP that reports on Java unchecked runtime exceptions. (true false)
isErrorPage	Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.
extends	Specifies a superclass that the generated servlet must extend
import	Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes.
info	Defines a string that can be accessed with the servlet's getServletInfo() method.
isThreadSafe	Defines the threading model for the generated servlet. (true false)
language	Defines the programming language used in the JSP page.
session	Specifies whether or not the JSP page participates in HTTP sessions
isELIgnored	Specifies whether or not EL expression within the JSP page will be ignored.
isScriptingEnabled	Determines if scripting elements are allowed for use.

You can write XML equivalent of the above syntax as follows:

```
<jsp:directive.page attribute="value" />
```

2) **include Directive:**

- **include** directive is used to includes a file during the translation phase.
- This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code *include* directives anywhere in your JSP page.

The general usage form of this directive is as follows:

```
<%@ include file="relative url" >
```

The filename in the include directive is actually a relative URL. If you just specify a filename with no associated path, the JSP compiler assumes that the file is in the same directory as your JSP.

You can write XML equivalent of the above syntax as follows:

```
<jsp:directive.include file="relative url" />
```

3) **The taglib Directive:**

- The JavaServer Pages API allows you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behaviour.
- The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.

The taglib directive follows the following syntax:

```
<%@ taglib uri="uri" prefix="prefixOfTag" >
```

Where the **uri** attribute value resolves to a location the container understands and the **prefix** attribute informs a container what bits of markup are custom actions.

You can write XML equivalent of the above syntax as follows:

```
<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />
```


[3] Implicit object

JSP Implicit Objects are the Java objects that the JSP Container makes available to developers in each page and developer can call them directly without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.

No.	Object	Description
1	request	This is the HttpServletRequest object associated with the request.
2	response	This is the HttpServletResponse object associated with the response to the client.
3	out	This is the PrintWriter object used to send output to the client.
4	session	This is the HttpSession object associated with the request.
5	application	This is the ServletContext object associated with application context.
6	config	This is the ServletConfig object associated with the page.
7	pageContext	This encapsulates use of server-specific features like higher performance JspWriters .
8	page	This is simply a synonym for this , and is used to call the methods defined by the translated servlet class.
9	Exception	The Exception object allows the exception data to be accessed by designated JSP.

❖ Create and execute JSP page

1) Create JSP Page

- To create the first jsp page, write some html code as given below, and save it by .jsp extension. We have save this file as index.jsp.
- Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the jsp page.

```

Web-app
|
demo
|
index.jsp

```

index.jsp

Let's see the simple example of JSP, here we are using the scriptlet tag to put java code in the JSP page. We will learn scriptlet tag later.

```
<html>
<body>
<% out.print(2*5); %>
</body>
</html>
```

It will print **10** on the browser.

2) Execute JSP Page

Follow the following steps to execute this JSP page:

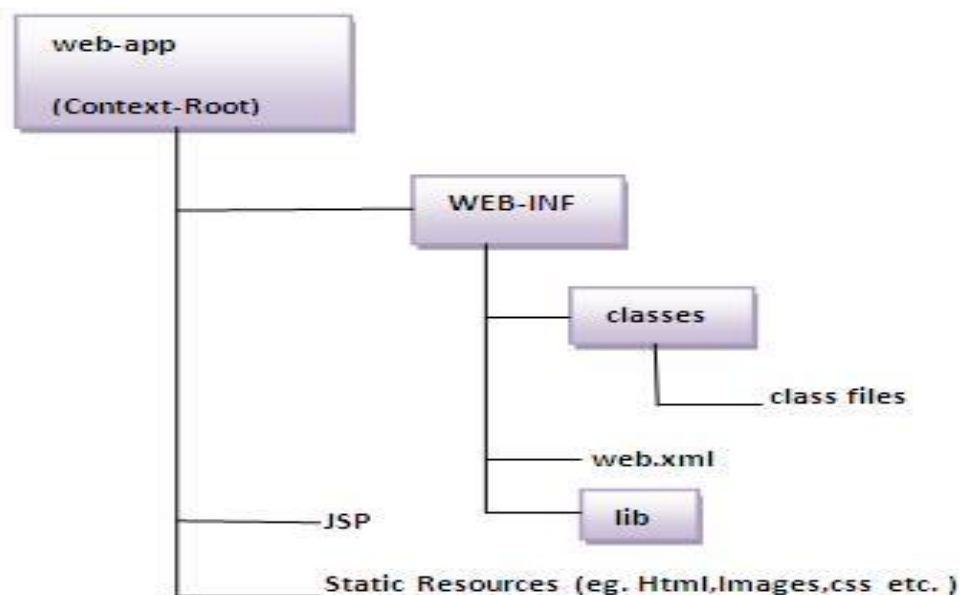
- Start the server
- put the .jsp file in a folder and deploy on the server
- visit the browser by the url `http://localhost:portno/contextRoot/jspfile` e.g.

`http://localhost:8080/demo/index.jsp`

- There is no need of directory structure if you don't have class files or tld files. For example, put jsp files in a folder directly and deploy that folder. It will be running fine. But if you are using bean class, Servlet or tld file then directory structure is required.

Directory structure of JSP

The directory structure of JSP page is same as servlet. We contains the jsp page outside the WEB-INF folder or in any directory.

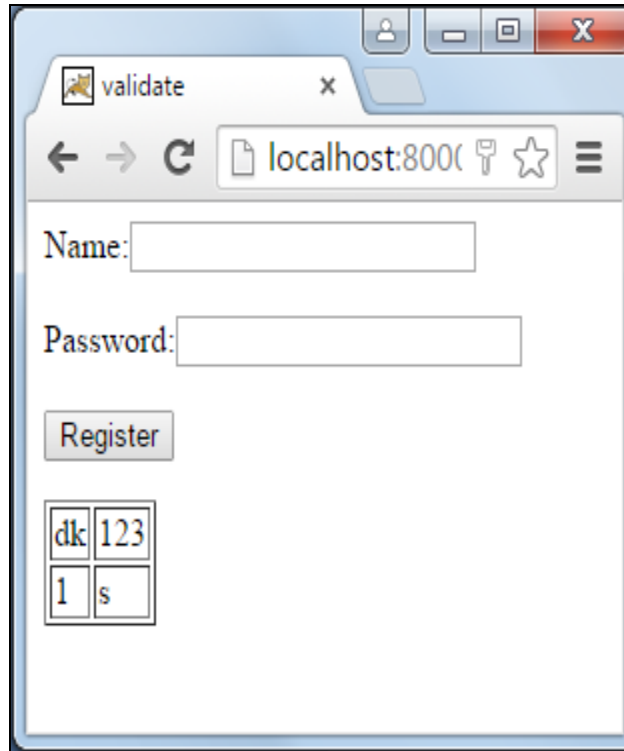


Example -9 :Write a JSP program to add new user in table using JDBC.**index.jsp**

```
<html>
  <head><title>validate</title></head>
<body>

<form action="Registration" method="get">
    Name:<input type="text"    name="userName"/>    <br/><br/>
    Password:<input type="password" name="userPass"/>    <br/><br/>
    <input type="submit"    value="Register"/>
</form>
<% @page import="java.sql.*"%>

    <%
    String uname = request.getParameter ("userName");
    String pass = request.getParameter ("userPass");
    try
    {
    Class.forName ("com.mysql.jdbc.Driver");
    Connection cn= DriverManager.getConnection ("jdbc:
mysql://localhost:3306/mydb","root","");
    Statement stmt = cn.createStatement ();
    ResultSet rs = stmt.executeQuery ("select * from user");
    out.print ("<table border='1'>");
    while (rs.next())
    {
        out.print("<tr>");
        out.print("<td>"    +rs.getString("username")    + "</td>");
        out.println("<td>" +rs.getString("password")    + "</td>");
        out.print("</tr>");
    }
    out.print("</table>");
    }
    catch(Exception e)
    {
    }
    %>
</body>
</html>
```



Example -10 Write a JSP program to calculate student grade.

index.jsp

```
<html>
  <head><title>PRACTICAL 18</title></head>
<body>
  <form action="StudentGrade" method="Post">
    <table>
      <tr>
        <td>Subject 1: </td>
        <td><input type="text" name="sub1"/></td>
      </tr>

      <tr>
        <td>Subject 2: </td>
        <td><input type="text" name="sub2"/></td>
      </tr>

      <tr>
        <td>Subject 3: </td>
        <td><input type="text" name="sub3"/></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

```
<tr>
    <td>Subject 4: </td>
    <td><input type="text" name="sub4"/></td>
</tr>
<tr>
    <td>Subject 5: </td>
    <td><input type="text" name="sub5"/></td>
</tr>
<tr><td><input type="submit" value="Submit"/></td>
</tr>
</table>
</form>
</body>
</html>
```

Grade.jsp

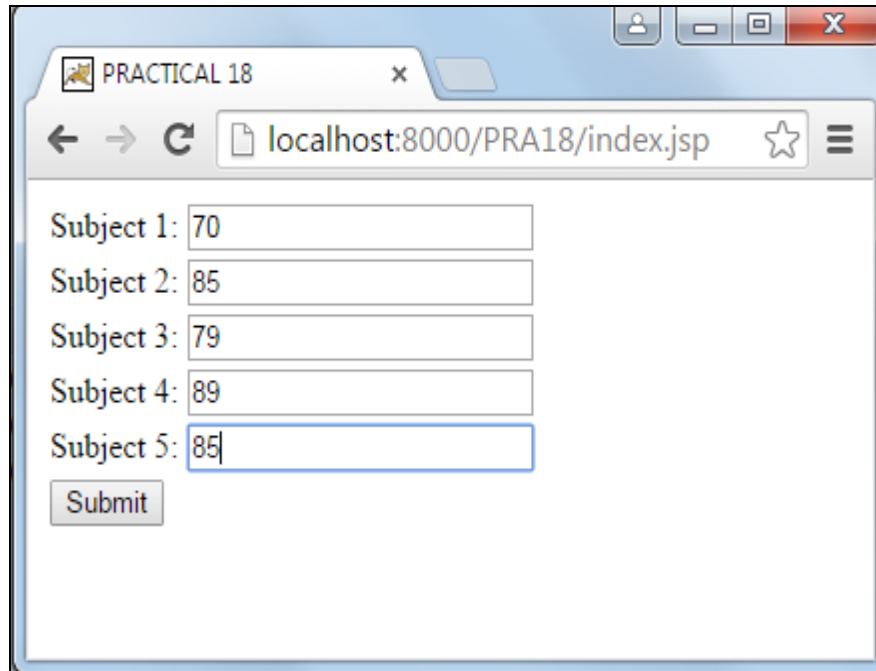
```
<html>
    <head><title>JSP Grade</title></head>
<body>
<%
    int sub1 = Integer.parseInt(request.getParameter("sub1"));
    int sub2 = Integer.parseInt(request.getParameter("sub2"));
    int sub3 = Integer.parseInt(request.getParameter("sub3"));
    int sub4 = Integer.parseInt(request.getParameter("sub4"));
    int sub5 = Integer.parseInt(request.getParameter("sub5"));

    int total = sub1 + sub2 + sub3 + sub4 + sub5;
    float per = (total * 100) / 500;

    out.println("Total = "+total+"<br />");
    out.println("Percentage = "+per+"<br />");

    if(per >= 75)
    {
        out.println("Distinction");
    }
    else if(per <= 74 && per >= 60)
    {
        out.println("First Class");
    }
    else if(per <= 59 && per >= 35)
```

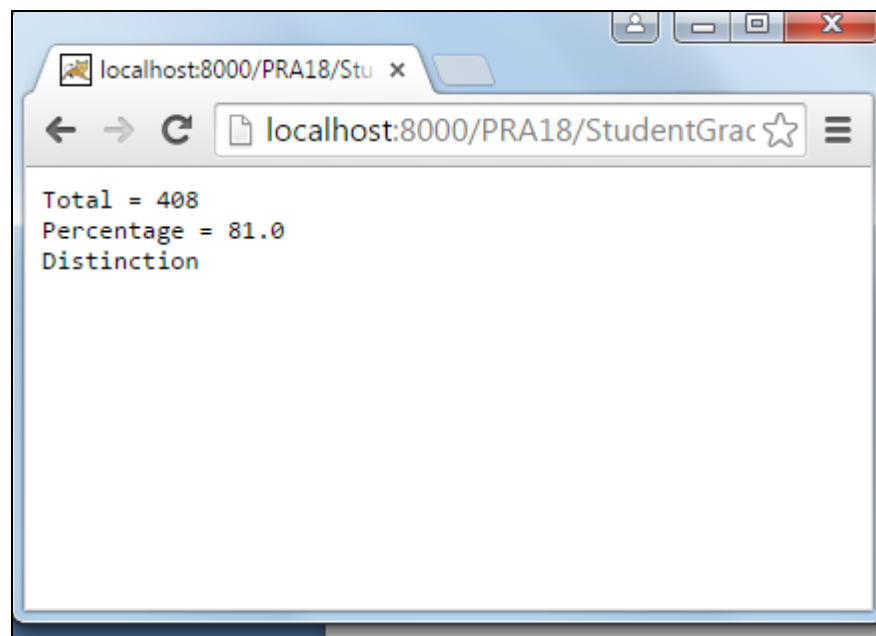
```
{
    out.println("Second Class");
}
else
{
    out.println("Fail");
}
%>
</body>
</html>
```



A screenshot of a web browser window titled "PRACTICAL 18". The address bar shows "localhost:8000/PRA18/index.jsp". The page contains five input fields labeled "Subject 1:", "Subject 2:", "Subject 3:", "Subject 4:", and "Subject 5:". The values entered are 70, 85, 79, 89, and 85 respectively. Below the input fields is a "Submit" button.

Subject	Mark
Subject 1:	70
Subject 2:	85
Subject 3:	79
Subject 4:	89
Subject 5:	85

Submit



A screenshot of a web browser window showing the result of the form submission. The address bar shows "localhost:8000/PRA18/StudentGrac". The page displays the following text:

Total = 408
Percentage = 81.0
Distinction

Example-11

Write a JSP program to create static login and dynamic login form for user.

Static login

```
web-app
|
PRA17
|-----index.jsp
|-----wel.jsp
```

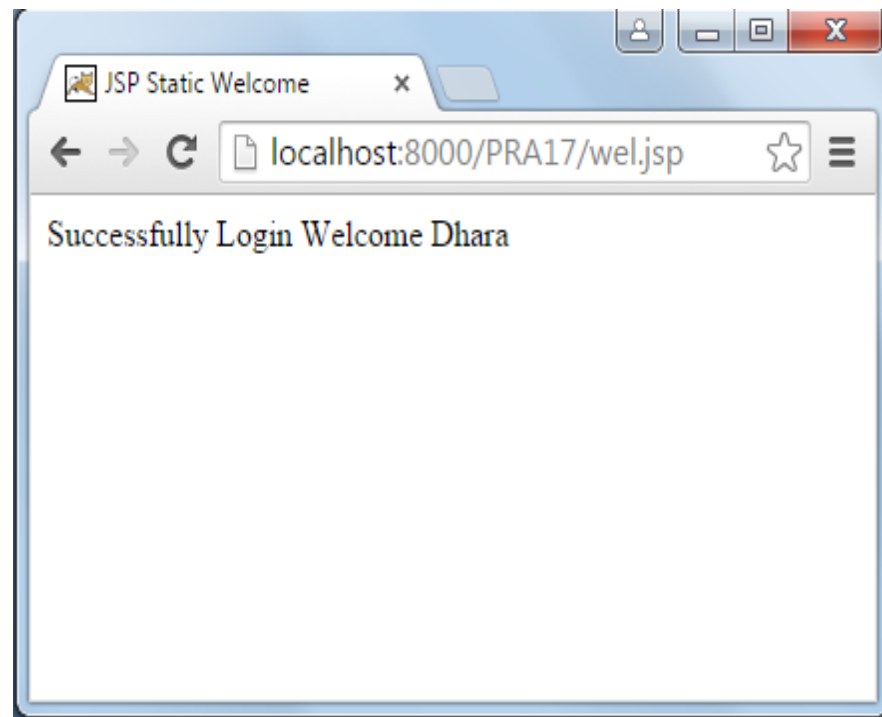
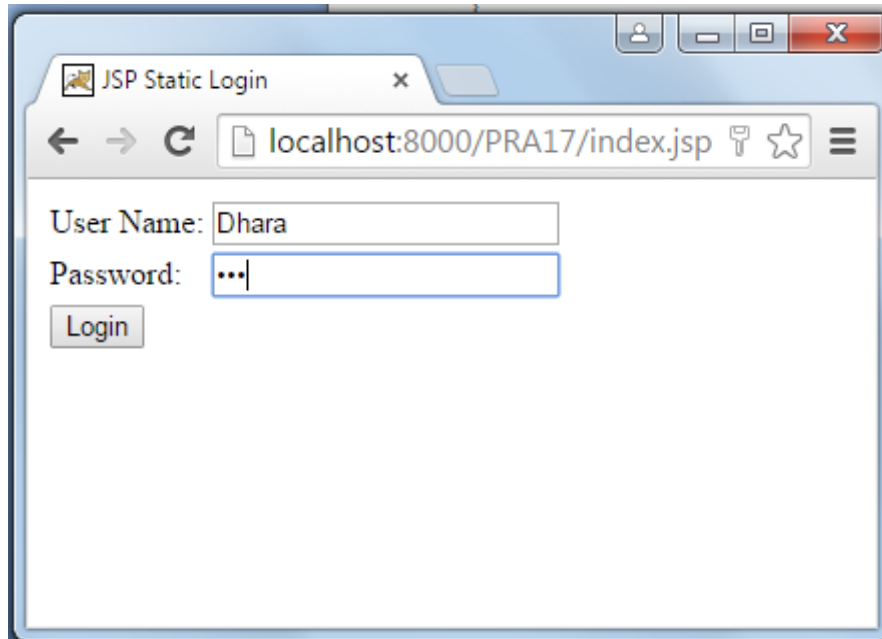
index.jsp

```
<html>
<head><title>JSP Static Login</title></head>
<body>
<form action="wel.jsp" method="Post">
<table>
  <tr>
    <td>User Name: </td>
    <td><input type="text" name="userName"/></td>
  </tr>
  <tr>
    <td>Password: </td>
    <td><input type="password" name="userPass"/></td>
  </tr>
  <tr><td><input type="submit" value="Login"/></td>
  </tr>
</table>
</form>
</body>
</html>
```

wel.jsp

```
<html>
<head><title>JSP Static Welcome</title></head>
<body>
<%
  String uname = request.getParameter("userName");
  String pass = request.getParameter("userPass");
  if(uname.equals("Dhara") && pass.equals("123"))
  {
    out.println("Successfully Login");
    out.println("Welcome "+uname);
  }
%>
```

```
    }  
    else  
    {  
        out.println("Sorry..!!! Wrong User Name & Password");  
    }  
%>  
</body>  
</html>
```



Dynamic login

```
web-app
|
PRA17
|-----dyndex.jsp
|-----dywel.jsp
```

dyindex.jsp

```
<html>
<head><title>JSP Dynamic Login</title></head>
<body>
<form action="DynamicLogin" method="Post">
<table>
  <tr>
    <td>User Name: </td>
    <td><input type="text" name="userName"/></td>
  </tr>

  <tr>
    <td>Password: </td>
    <td><input type="password" name="userPass"/></td>
  </tr>
  <tr><td><input type="submit" value="Login"/></td>
  </tr>
</table>
</form>
</body>
</html>
```

dywel.jsp

```
<%@page import="java.sql.*"%>
<html>
<head><title>JSP Dynamic Welcome</title></head>
<body>
<%
  String uname = request.getParameter("userName");
  String pass = request.getParameter("userPass");
  try
  {
    Class.forName("com.mysql.jdbc.Driver");
    Connection cn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb","root","");
```

```
Statement stmt = cn.createStatement();
ResultSet rs = stmt.executeQuery("select * from login where username
= '"+uname+"' and password = '"+pass+"'");

    if(rs.next())
    {
        out.println("Successfully Login");
        out.println("Welcome "+uname);
    }
    else
    {
        out.println("Sorry..!!! Wrong User Name & Password");
    }

}
catch(Exception e)
{
    out.print(e);
}
%>
</body>
</html>
```

