# SESSION-11

# JSP (JAVA SERVER PAGES)

JSP is the technology and whose specification is implemented by server vendors. JSP is an alternative technology for servlets. Since, servlets having the following limitations:

**Phases in JSP:**

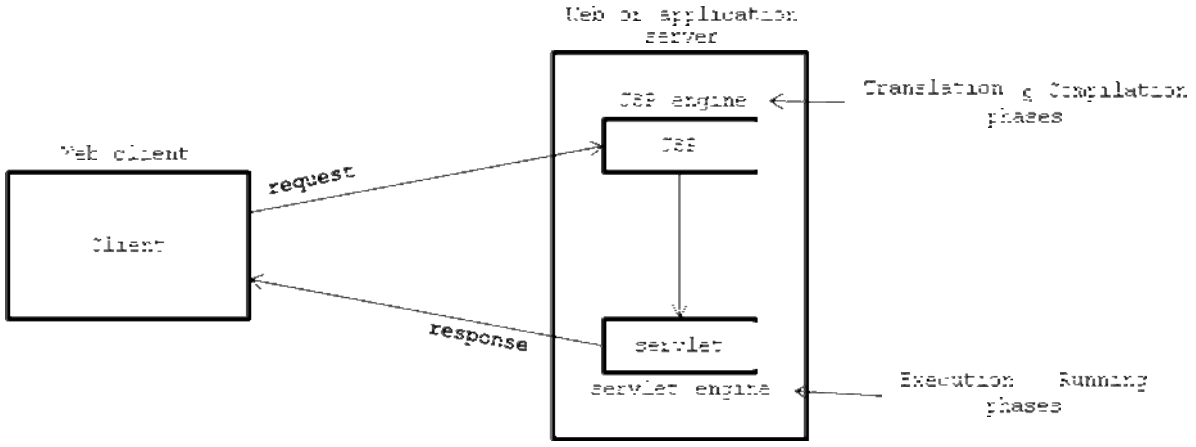
Whenever we write in a JSP page, that JSP page will undergo the following phases:

1. Translation phase: It is one which converts. jsp program into .java program internally by the container. Once the translation phase is completed the entire JSP program is available into a pure java program.
2. Compilation phase: It is one which converts .java program into .class file provided no errors found in .java program by the container. If errors are found by the container in .java program those errors will be listed in the browser.
3. Execution or Running phase: It is the process of executing .class file by the container.

| Limitations of servlets | JSP provides |
|---|---|
| 1. In order to develop any servlet, they must know java language. | 1. To write any JSP program java language is not necessary i.e., a non-java programmer can write the JSP page effectively. |
| 2. Servlets provides uncomforting. Since, in a single servlet we are writing presentation logic and application or business logic. | 2. In JSP there is a separation between presentation logic and application or business logic. |
| 3. Maintenance and deployment problems are more (servlets allows only static changes). | 3. JSP page can minimize or reduce maintenance and deployment problems. Since, it allows dynamic changes. |
| 4. Important: Servlets does not provide automatic page compilation i.e., every servlet program must be compiled explicitly. | 4. JSP provides automatic page compilation (every JSP program internally translated as a java program by the container which is nothing but servlet). |
| 5. Servlets does not provide any custom tag (user defined tag) development. | 5. JSP provides the features to develop custom tags. |
| 6. Servlets does not provide any implicit objects. | 6. Every JSP page contains 'n' number of implicit objects such as out, session, exception, etc. |

When we make very first request to a JSP page, translation phase, compilation phase and execution phase will be taken place. From second request to further sub sequent requests only execution phase taking place provided when we are not modifying JSP page.

**JSP architecture:**



# Life cycle methods of JSP:

Since, JSP is the server side dynamic technology and it extends the functionality of web or application server, it contains the following life cycle methods:

public void jspInit ();

public void jspService (ServletRequest, ServletResponse);

public void jspDestroy ();

The above three life cycle methods are exactly similar to life cycle methods of servlet.

TAGS in JSP

Writing a program in JSP is nothing but making use of various tags which are available in JSP. In JSP we have three categories of tags; they are scripting elements, directives and standard actions.

## SCRIPTING ELEMENTS:

*Scripting elements* are basically used to develop preliminary programming in JSP such as, declaration of variables, expressions and writing the java code. Scripting elements are divided into three types; they are declaration tag, expression tag and scriplet.

### 1. Declaration tag:

Whenever we use any variables as a part of JSP we have to use those variables in the form of declaration tag i.e., declaration tag is used for declaring the variables in JSP page.

Syntax:

```
<%! Variable declaration or method definition %>
```

When we declare any variable as a part of declaration tag those variables will be available as data members in the servlet and they can be accessed through out the entire servlet.

When we use any methods definition as a part of declaration tag they will be available as member methods in servlet and it will be called automatically by the servlet container as a part of service method.

**For example-1:**

```
<%!    int a=10, b=30, c;        %>
```

**For example-2:**

```
<%!
       int count ()
       {
              return (a+b);
       }
%>
```

### 2. Expression tag:

Expression tags are used for writing the java valid expressions as a part of JSP page.

Syntax:

```
<%= java valid expression %>
```

Whatever the expression we write as a part of expression tags that will be given as a response to client by the servlet container. All the expression we write in expression tag they

will be placed automatically in `out.println ()` method and this method is available as a part of service method.

NOTE: Expressions in the expression tag should not be terminated by semi-colon (;) .

For example-1:

```
<%! int a=10, b=20 %>
<%= a+b %>
```

The equivalent servlet code for the above expression tag is `out.println (a+b);` out is implicit object of *JSPWriter* class.

**For example-2:**

`<%= new java.util.Date () %>` € `out.println (new java.util.Date ());`

## 3. Scriplet tag:

Scriplets are basically used to write a pure java code. Whatever the java code we write as a part of scriplet, that code will be available as a part of `service ()` method of servlet.

Syntax:

```
<% pure java code %>
```

**Write a scriplet for generating current system date?**

Answer:

**web.xml:**

```
<web-apps>
</web-apps>
```

**DateTime.java:**

```
<html>
	<title>Current Date & Time</title>
	<head><h4>Current date & time</h4></head>
	<body>
		<%
			Date d=new Date ();
			String s=d.toString ();
			out.println (s);
		%>
	</body>
</html>
```

[or]

```
<html>


	<title>Current Date & Time</title>
	<head><h4>Current date & time</h4></head>
	<body>
		<%=new Date ()%>
	</body>
</html>
```

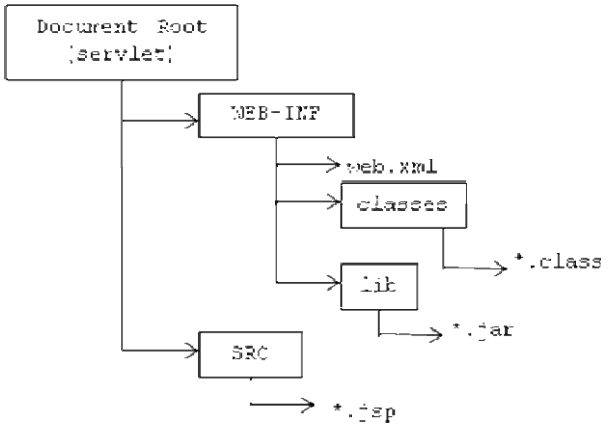
**Write a JSP page to print 1 to 10 numbers? [For *web.xml* refer page no: 102]** Answer:

One2TenNumbers.jsp:

```
<html>
	<title>Print Numbers 1-10</title>
	<head>Numbers 1-10</head><br>
```

```
	<body>
		<%
			for (int i=1; i<=10; i++)
			{
				out.println (i);
			}
		%>
	</body>
</html>
```

**In order to execute any JSP program one must follow the following directory structure:**




**Write JSP program to print "Hello JSP world to KALYAN"? [For *web.xml* refer page no: 102]** Answer:

HelloJSP.jsp:

```
<html>
	<title>Fisrt Trail</title>
	<head>Fresher to JSP</head><br>
	<body>
		<h3>Hello JSP world to KALYAN</h3>
	</body>
</html>
```

**NOTE:**

Whenever we deploy a JSP application in webapps folder of Tomcat we get an appropriate equivalent servlet for the corresponding JSP file. For example, when we deploy first.jsp through a *document root* first in webapps folder of tomcat we get first_jsp.java (which is nothing but a servlet) and first_jsp.class by Tomcat server.

The location of servlet and .class file is as follows:

**Write a JSP page which will display current data and time? [For *web.xml* refer page no: 102]** Answer:

DateTime.jsp:

```
<html>
	<title>Current Date & Time</title>
	<head>Date & Time without using out.println</head><br>
	<body>
		<%
```

```
                java.util.Date d=new java.util.Date ();
            %>
            <h4>Current date & time</h4>
            <h3><%= d %></h3>
      </body>
</html>
```

**Write a JSP page which will display number of times a request is made [write a JSP for hit counter]? [For *web.xml* refer page no: 102]**

Answer:

ReloadPageCount.jsp:

```
<html>
      <title>Number of Reloads</title>
      <head>Number of visitings to a browser</head><br>
      <body>
            <%! int ctr=0; %>
            <%!
                  int count ()
                  {
                        return (++ctr);
                  }
            %>
            <h3><%= count () %></h3>
      </body>
</html>
```

NOTE:

Within servlet we use to write html code to generate presentation logic whereas in JSP environment within html program we are making use of JSP tags.

Write a JSP page which will retrieve the data from database? [For *web.xml* refer page no: 102]

Answer:

```
<%@ page import="java.sql.*, java.io.*" %>
<html>
  <title>Data From Database</title>
  <head>Retrieve data from Datebase</head>
  <body>
      <%!
        Connection con=null;
        Statement st=null;
        public void jspInit ()
        {
            try
            {
              Class.forName ("oracle.jdbc.driver.OracleDriver");
              con=DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:Hanuman","scott",
"tiger");
              st=con.createStatement ();
            }
            catch (Exception e)
            {
              out.println (e);
            }
        }
```

```
%>
<%!
  try
  {
       ResultSet rs=st.executeQuery ("select * from employee");
       while (rs.next ())
       {
```

```
                out.println ("<h3>"+rs.getString (1)+"          "+rs.getString (2)+"</h3>")
                }
            }
            catch (Exception e)
            {
                out.println (e);
            }
        %>
    </body>
</html>
```