Java Server Page (JSP)

What is JSP?

JSP technology is used to create web application 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.

Advantages of JSP

1) Extension to Servlet

2) Easy to maintain

3) Fast Development: No need to recompile and redeploy

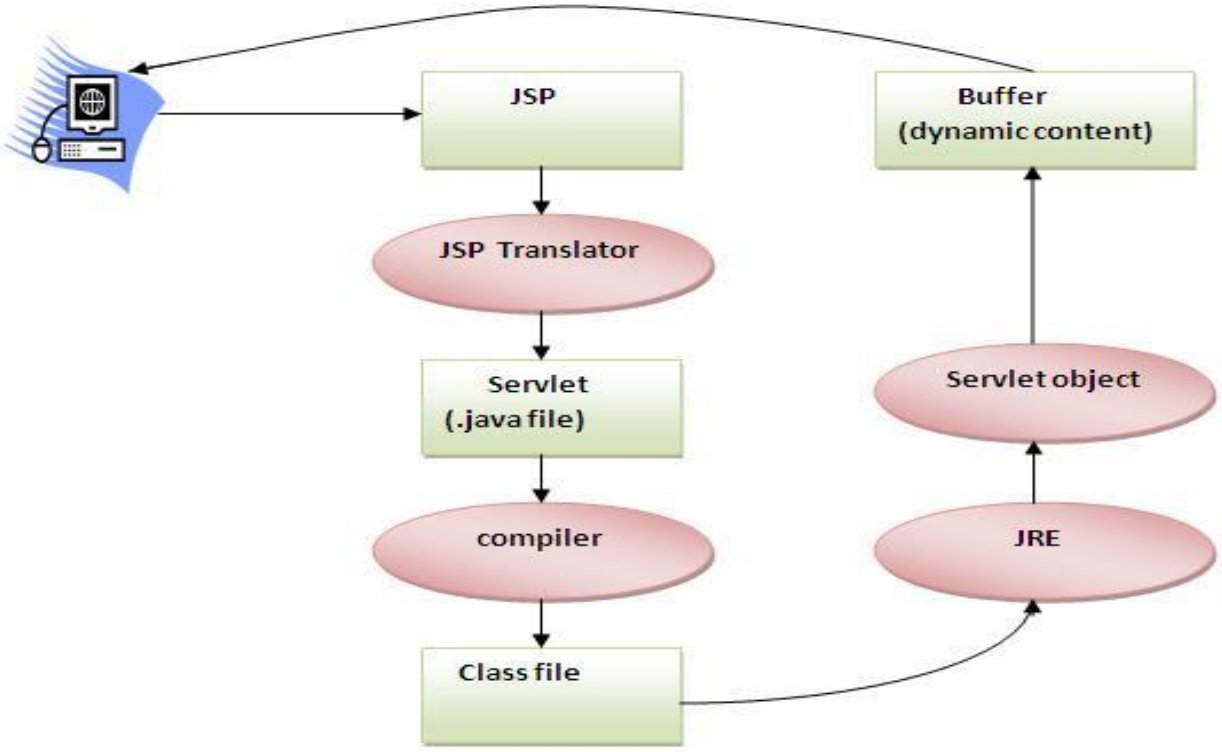
4) Less code than Servlet

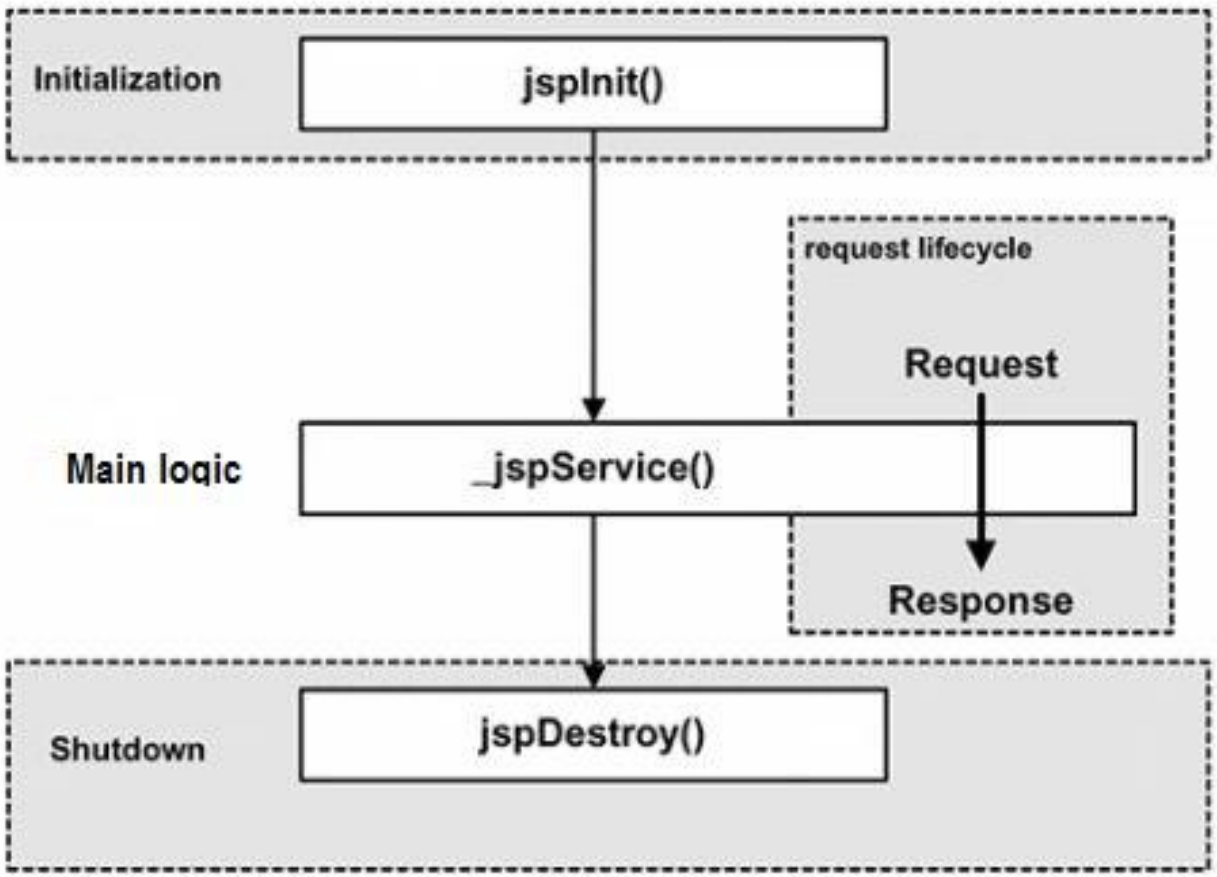
Life Cycle of a JSP Page

The JSP page follows these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Class loading (class file is loaded by the class loader)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Request processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

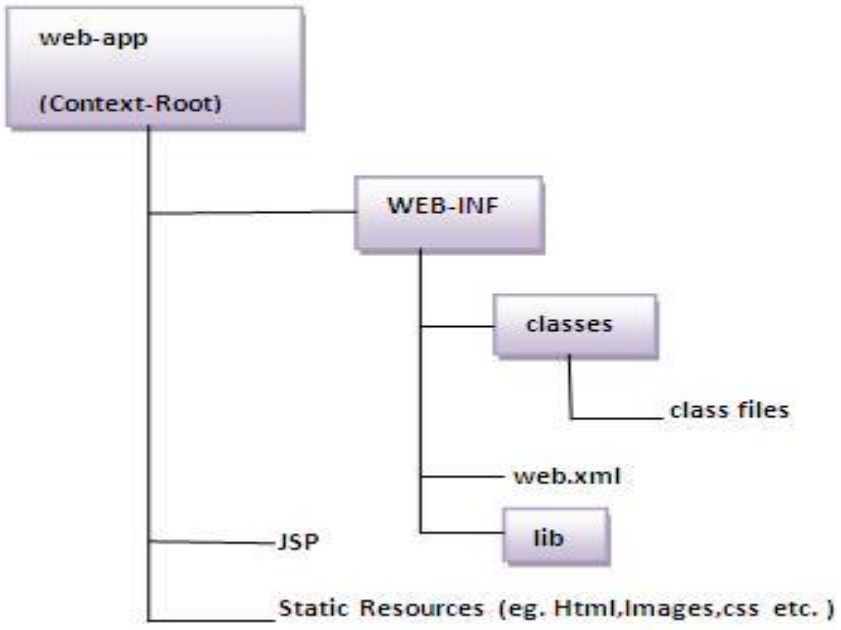
Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.


JSP
Buffer
(dynamic content)
JSP Translator
Servlet
(.java file)
Servlet object
compiler
JRE
Class file


Initialization
jspInit()
request lifecycle
Request
Main logic
_jspService()
Response
Shutdown
jspDestroy()

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.



Scripting Elements

The scripting element provides the ability to insert java code inside the JSP. There are three types of scripting elements:

- Scriptlet tag
- Expression tag
- Declaration tag

JSP Scriptlet Tag

A Sriptlet tag is used to execute java source code in JSP. Syntax is as follows:

<% java source code %>

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

```
<jsp:scriptlet>
  code fragment
</jsp:scriptlet>
```

Simple Example of JSP Scriptlet Tag

In this example, we are displaying a welcome message.

```
1. <html>
2. <body>
3. <% out.print("welcome to jsp"); %>
4. </body>
5. </html>
```

Example of JSP Scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%
4. String name=request.getParameter("uname");
5. out.print("welcome "+name);
6. %>
7. </form>
8. </body>
9. </html>
```

JSP Expression Tag

The code placed within expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP Expression Tag

```
<%= statement %>
```

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

```
<jsp:expression>
  expression
</jsp:expression>
```

Example of JSP Expression Tag

In this example of JSP expression tag, we are simply displaying a welcome message.

```
1. <html>
2. <body>
3. <%= "welcome to jsp" %>
4. </body>
5.
6. </html>
```

Note: Do not end your statement with semicolon in case of expression tag.

Example of JSP expression tag that prints current time

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

index.jsp

```
1. <html>
2. <body>
3. Current Time: <%= java.util.Calendar.getInstance().getTime() %>
4. </body>
5. </html>
```

Example of JSP expression tag that prints the user name

In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username.

index.html

```
1. <html>
2. <body>
3.
4. <form action="welcome.jsp">
5. <input type="text" name="uname"><br/>
6. <input type="submit" value="go">
7. </form>
8. </body>
9. </html>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%= "Welcome "+request.getParameter("uname") %>
4. </form>
5. </body>
6. </html>
```

JSP Declaration Tag

The JSP declaration tag is used to declare fields and methods. The code written inside the JSP declaration tag is placed outside the service() method of auto generated Servlet. So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

```
<%!  field or method declaration  %>
```

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

```
<jsp:declaration>
  code fragment
</jsp:declaration>
```

Difference between the JSP scriptlet tag and JSP declaration tag ?

JSP Scriptlet Tag	JSP Declaration Tag
TheJSP scriptlet tag can only declare variables not methods.	The JSP declaration tag can declare variables as well as methods.
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of JSP declaration tag is placed outside the _jspService() method.

Example of JSP declaration tag that declares field

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the JSP expression tag.

index.jsp

```
1. <html>
2. <body>
3.
4. <%! int data=50; %>
5. <%= "Value of the variable is:"+data %>
6.
7. </body>
8. </html>
```

Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the JSP expression tag. But we can also use JSP scriptlet tag to call the declared method.

index.jsp

```
1. <html>
2. <body>
3.
4. <%!
5. int cube(int n){
6. return n*n*n*;
7. }
8. %>
9.
10.        <%= "Cube of 3 is:"+cube(3) %>
11.
```

```
12.     </body>
13.     </html>
```

JSP Implicit Objects

There are 9 JSP implicit objects. These objects are created by the web containers that are available to all the JSP pages. The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

config implicit object

In JSP, config is an implicit object of type ServletConfig. This object can be used to get configuration information for a particular JSP page. This variable information can be used for one JSP page only.

Example of config implicit object:

index.html

```
1. <html>
2. <body>
3. <form action="welcome">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

web.xml file

```
1. <web-app>
2.
3. <servlet>
4. <servlet-name>sonoojaiswal</servlet-name>
5. <jsp-file>/welcome.jsp</jsp-file>
6.
7. <init-param>
8. <param-name>dname</param-name>
9. <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
10.     </init-param>
11.
12.     </servlet>
13.
14.     <servlet-mapping>
15.     <servlet-name>sonoojaiswal</servlet-name>
16.     <url-pattern>/welcome</url-pattern>
17.     </servlet-mapping>
18.
19.     </web-app>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%
4.
5. out.print("Welcome "+request.getParameter("uname"));
6.
7. String driver=config.getInitParameter("dname");
8. out.print("driver name is="+driver);
9.
10.
11.     %>
12.     </body>
13.     </html>
```

pageContext implicit object

In JSP, pageContext is an implicit object of type PageContext class. The pageContext object can be used to set, get or remove attribute from one of the following scopes:

- page
- request
- session
- application

In JSP, page scope is the default scope.

Example of pageContext implicit object

index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%
4.
5. String name=request.getParameter("uname");
6. out.print("Welcome "+name);
7.
8. pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);
9.
10.        <a href="second.jsp">second jsp page</a>
11.
12.        %>
13.        </body>
14.        </html>
```

second.jsp

```
1. <html>
2. <body>
3. <%
4.
5. String name=(String)pageContext.getAttribute("user",PageContext.SESSION_SCO
   PE);
6. out.print("Hello "+name);
7.
8. %>
9. </body>
10.       </html>
```

JSP directives

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

- page directive
- include directive
- taglib directive

Syntax of JSP Directive

<%@ directive attribute="value" %>

JSP page directive

The page directive defines attributes that apply to an entire JSP page.

Syntax of JSP page directive

<%@ page attribute="value" %>

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored
- isThreadSafe
- autoFlush

- session
- pageEncoding
- errorPage
- isErrorPage

---

1) import

The import attribute is used to import class, interface or all the members of a package. It is similar to import keyword in java class or interface.

Example of import attribute

```
1. <html>
2. <body>
3.
4. <%@ page import="java.util.Date" %>
5. Today is: <%= new Date() %>
6.
7. </body>
8. </html>
```

2) contentType

The contentType attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.The default value is "text/html;charset=ISO-8859-1".

Example of contentType attribute

```
1. <html>
2. <body>
3.
4. <%@ page contentType=application/msword %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

3) extends

The extends attribute defines the parent class that will be inherited by the generated

servlet. It is rarely used.

4) info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

Example of info attribute

```
1. <html>
2. <body>
3.
4. <%@ page info="composed by Sonoo Jaiswal" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

The web container will create a method getServletInfo() in the resulting servlet. For example:

```
1. public String getServletInfo() {
2.   return "composed by Sonoo Jaiswal";
3. }
```

5) buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page. The default size of the buffer is 8Kb.

Example of buffer attribute

```
1. <html>
2. <body>
3.
4. <%@ page buffer="16kb" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

6) language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

7) isThreadSafe

Servlet and JSP both are multithreaded.If you want to control this behaviour of JSP page, you can use isThreadSafe attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.If you make the value of isThreadSafe attribute like:

```
<%@ page isThreadSafe="false" %>
```

The web container in such a case, will generate the servlet as:

```
1. public class SimplePage_jsp extends HttpJspBase
2.   implements SingleThreadModel{
3. .......
4. }
```

8) errorPage

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

Example of errorPage attribute

```
1. //index.jsp
2. <html>
3. <body>
4.
5. <%@ page errorPage="myerrorpage.jsp" %>
6.
7.  <%= 100/0 %>
8.
9. </body>
10.       </html>
```

9) isErrorPage

The isErrorPage attribute is used to declare that the current page is the error page.

Note: The exception object can only be used in the error page.

Example of isErrorPage attribute

```
1. //myerrorpage.jsp
2. <html>
3. <body>
4.
5. <%@ page isErrorPage="true" %>
6.
7.  Sorry an exception occured!<br>
8. The exception is: <%= exception %>
9.
10.        </body>
11.        </html>
```

Include directive

The include directive is used to include the contents of any resource it may be JSP file, html file or text file. The include directive includes the original content of the included resource at page translation time (the JSP page is translated only once so it will be better to include static resource).

Advantage of Include directive

Code Reusability

Syntax of include directive

```
1. <%@ include file="resourceName" %>
```

Example of include directive

In this example, we are including the content of the header.html file. To run this example you must create an header.html file.

```
1. <html>
2. <body>
3.
4. <%@ include file="header.html" %>
5.
6. Today is: <%= java.util.Calendar.getInstance().getTime() %>
7.
```

8. </body>
9. </html>

JSP Action Tags (Action Elements)

There are many JSP action tags or elements. Each tag is used to perform some specific tasks. The action tags basically are used to control the flow between pages and to use Java Bean. Jsp action tags are as follows:

- jsp:forward
- jsp:include
- jsp:useBean
- jsp:setProperty
- jsp:getProperty
- jsp:plugin
- jsp:param
- jsp:fallback

The jsp:useBean, jsp:setProperty and jsp:getProperty tags are used for bean development. So we will see these tags in bean developement.

---

jsp:forward action tag

The jsp:forward action tag is used to forward the request to another resource it may be jsp, html or another resource.

Syntax of jsp:forward action tag without parameter

1. <jsp:forward page="relativeURL | <%= expression %>" />

Syntax of jsp:forward action tag with parameter

1. <jsp:forward page="relativeURL | <%= expression %>">
2. <jsp:param name="parametername" value="parametervalue | <%=expression% >" />
3. </jsp:forward>

Example of jsp:forward action tag without parameter

In this example, we are simply forwarding the request to the printdate.jsp file.

index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" />
</body>
</html>
```

printdate.jsp

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

Example of jsp:forward action tag with parameter

In this example, we are forwarding the request to the printdate.jsp file with parameter and printdate.jsp file prints the parameter value with date and time.

index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" >
<jsp:param name="name" value="javatpoint.com" />
</jsp:forward>

</body>
        </html>
```

printdate.jsp

```
<html>
<body>

<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
<%= request.getParameter("name") %>

</body>
</html>
```

jsp:include action tag

The **jsp:include action tag** is used to include the content of another resource it may be jsp, html or servlet.

The jsp include action tag includes the resource at request time so it is **better for dynamic pages** because there might be changes in future.

Advantage of jsp:include action tag

code reusability

Syntax of jsp:include action tag without parameter

```
<jsp:include page="relativeURL | <%= expression %>" />
```

Syntax of jsp:include action tag with parameter

```
<jsp:include page="relativeURL | <%= expression %>">
<jsp:param name="parametername" value="parametervalue | <%=expression% >" />
</jsp:include>
```

Example of jsp:include action tag without parameter

In this example, index.jsp file includes the content of the printdate.jsp file.

File: index.jsp

```
<html>
<body>
<h2>this is index page</h2>

<jsp:include page="printdate.jsp" />

<h2>end section of index page</h2>
</body>
</html>
```

File: printdate.jsp

```
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
```

Setting Cookies with JSP

HTML File

```
<html>
<body>
<form action="main.jsp" method="GET">
First Name: <input type="text" name="first_name">
<br />
Last Name: <input type="text" name="last_name" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

main.jsp

```
<%
   Cookie firstName = new Cookie("first_name", request.getParameter("first_name"));
   Cookie lastName = new Cookie("last_name", request.getParameter("last_name"));

   // Set expiry date after 24 Hrs for both the cookies.
   firstName.setMaxAge(60*60*24);
   lastName.setMaxAge(60*60*24);

   response.addCookie( firstName );
   response.addCookie( lastName );
%>
<html>
<body>
<center>
<h1>Setting Cookies</h1>
</center>
<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>
```

Reading Cookies with JSP

```
<html>
<body>
<center>
<h1>Reading Cookies</h1>
</center>
<%
   Cookie cookie = null;
   Cookie[] cookies = null;
   // Get an array of Cookies associated with this domain
   cookies = request.getCookies();
   if( cookies != null ){
      out.println("<h2> Found Cookies Name and Value</h2>");
      for (int i = 0; i < cookies.length; i++){
         cookie = cookies[i];
         out.print("Name : " + cookie.getName( ) + ",  ");
         out.print("Value: " + cookie.getValue( )+" <br/>");
      }
  }else{
      out.println("<h2>No cookies founds</h2>");
  }
%>
</body>
</html>
```

Delete Cookies with JSP

To delete cookies is very simple. If you want to delete a cookie then you simply need to follow up following three steps:

- Read an already existing cookie and store it in Cookie object.
- Set cookie age as zero using **setMaxAge()** method to delete an existing cookie.
- Add this cookie back into response header.

Example:

```
<html>
<body>
<center>
<h1>Reading Cookies</h1>
</center>
<%
   Cookie cookie = null;
   Cookie[] cookies = null;
   // Get an array of Cookies associated with this domain
   cookies = request.getCookies();
   if( cookies != null ){
      out.println("<h2> Found Cookies Name and Value</h2>");
      for (int i = 0; i < cookies.length; i++){
```

```
        cookie = cookies[i];
        if((cookie.getName( )).compareTo("first_name") == 0 ){
           cookie.setMaxAge(0);
           response.addCookie(cookie);
           out.print("Deleted cookie: " +
           cookie.getName( ) + "<br/>");
        }
        out.print("Name : " + cookie.getName( ) + ",  ");
        out.print("Value: " + cookie.getValue( )+" <br/>");
      }
  }else{
      out.println(
      "<h2>No cookies founds</h2>");
  }
%>
</body>
</html>
```

HttpSession Object

PageCounter Example

```
<%@ page import="java.io.*,java.util.*" %>
<%
   // Get session creation time.
   Date createTime = new Date(session.getCreationTime());
   // Get last access time of this web page.
   Date lastAccessTime = new Date(session.getLastAccessedTime());

   String title = "Welcome Back to my website";
   Integer visitCount = new Integer(0);
   String visitCountKey = new String("visitCount");
   String userIDKey = new String("userID");
   String userID = new String("ABCD");

   // Check if this is new comer on your web page.
   if (session.isNew()){
      title = "Welcome to my website";
      session.setAttribute(userIDKey, userID);
      session.setAttribute(visitCountKey,  visitCount);
   }
   visitCount = (Integer)session.getAttribute(visitCountKey);
   visitCount = visitCount + 1;
   userID = (String)session.getAttribute(userIDKey);
   session.setAttribute(visitCountKey,  visitCount);
%>
<html>
<head>
```

```
<title>Session Tracking</title>
</head>
<body>
<center>
<h1>Session Tracking</h1>
</center>
<table border="1" align="center">
<tr bgcolor="#949494">
   <th>Session info</th>
   <th>Value</th>
</tr>
<tr>
   <td>id</td>
   <td><% out.print( session.getId()); %></td>
</tr>
<tr>
   <td>Creation Time</td>
   <td><% out.print(createTime); %></td>
</tr>
<tr>
   <td>Time of Last Access</td>
   <td><% out.print(lastAccessTime); %></td>
</tr>
<tr>
   <td>User ID</td>
   <td><% out.print(userID); %></td>
</tr>
<tr>
   <td>Number of visits</td>
   <td><% out.print(visitCount); %></td>
</tr>
</table>
</body>
</html>
```

Database Operation in JSP

SELECT Operation

Following example shows how we can execute SQL SELECT statement using JTSL in JSP programming:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>
<head>
```

```
<title>SELECT Operation</title>
</head>
<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>

<sql:query dataSource="${snapshot}" var="result">
SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th>
   <th>First Name</th>
   <th>Last Name</th>
   <th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table>

</body>
</html>
```

INSERT Operation

Following example shows how we can execute SQL INSERT statement using JTSL in JSP programming:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>
<head>
<title>JINSERT Operation</title>
</head>
<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
```

```
    url="jdbc:mysql://localhost/TEST"
    user="root"  password="pass123"/>


<sql:update dataSource="${snapshot}" var="result">
INSERT INTO Employees VALUES (104, 2, 'Nuha', 'Ali');
</sql:update>

<sql:query dataSource="${snapshot}" var="result">
SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th>
   <th>First Name</th>
   <th>Last Name</th>
   <th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table>

</body>
</html>
```

DELETE Operation

Following example shows how we can execute SQL DELETE statement using JTSL in JSP programming:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>
<head>
<title>DELETE Operation</title>
</head>
<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
```

```
    url="jdbc:mysql://localhost/TEST"
    user="root"  password="pass123"/>

<c:set var="empId" value="103"/>

<sql:update dataSource="${snapshot}" var="count">
  DELETE FROM Employees WHERE Id = ?
  <sql:param value="${empId}" />
</sql:update>

<sql:query dataSource="${snapshot}" var="result">
   SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th>
   <th>First Name</th>
   <th>Last Name</th>
   <th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table>

</body>
</html>
```

UPDATE Operation

Following example shows how we can execute SQL UPDATE statement using JTSL in JSP programming:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>
<head>
<title>DELETE Operation</title>
</head>
<body>
```

```
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/TEST"
     user="root"  password="pass123"/>

<c:set var="empId" value="102"/>

<sql:update dataSource="${snapshot}" var="count">
  UPDATE Employees SET last = 'Ali'
  <sql:param value="${empId}" />
</sql:update>

<sql:query dataSource="${snapshot}" var="result">
   SELECT * from Employees;
</sql:query>

<table border="1" width="100%">
<tr>
   <th>Emp ID</th>
   <th>First Name</th>
   <th>Last Name</th>
   <th>Age</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
   <td><c:out value="${row.id}"/></td>
   <td><c:out value="${row.first}"/></td>
   <td><c:out value="${row.last}"/></td>
   <td><c:out value="${row.age}"/></td>
</tr>
</c:forEach>
</table>
 </body>
</html>
```