JSP

Java Server Pages

- Satya Kaveti

Small Codes

Programming Simplified

A SmlCodes.Com Small presentation

In Association with Idleposts.com

JSP by Satya Kaveti

First published on 3rd Nov 2016, Published by **SmlCodes.com**

Author Credits

Name : **Satya Kaveti**

Email : satyakaveti@gmail.com

Website : smlcodes.com, satyajohnny.blogspot.com

Digital Partners

Table of Contents

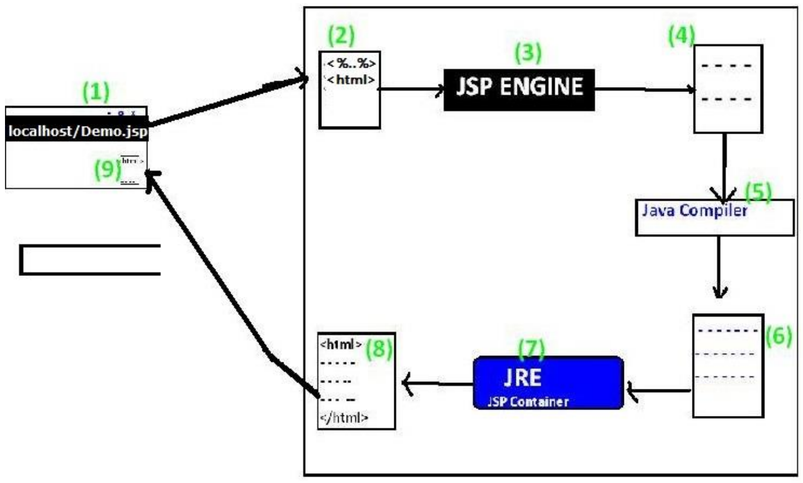
1. JSP Introduction 4
2. JSP Scriptlets 5
3. JSP Implicit objects 6
4. JSP Directives 8
5. JSP Action Tags 11
6. JSP EL (JSP Expression Language) 13
7. JSP - Standard Tag Library (JSTL) 14
8. JSP Custom Tags 16

1. JSP Introduction

Features of JSP

- Extension to Servlet
- Easy to maintain
- Fast Development: No need to recompile and redeploy
- Less code than Servlet

JSP Architecure



1.Clinet sends the Rqst from Browser for Demo.jsp

2.JSP is Deployed in server, that is Loaded

3.JSP Engine Converts JSP code [html,DHTML,java script] into appropriate 'SERVLET' code

4.'.java' of Servlet Contains Converted code of JSP Tags, and <html> just palced as same.Bcoz Java Compiler cannot under stand <HTML> code

5."JavaCompiler" takes that code and Exicuted

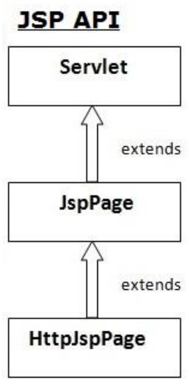
6.'.java' is Converted into ".class" file

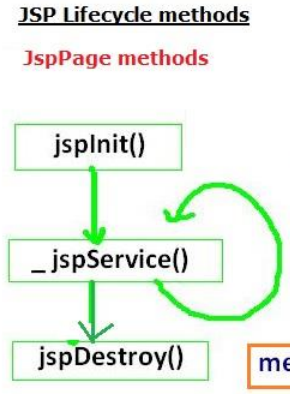
7.that '.class' is Taken by JRE and Exicuted

8.That JRE gives Output as <HTML> code, and send to clint

9.the <html> code is exicuted by Browser,and gives Output

JSP Lifecycle





jspInit()

- this is exceuted only once, when is 1st Rqst comes.
- It can be overridden
- JDBC Connction logic, one time executon logic we will write here

_jspService

- this is exceuted for Many times
- It cannot be overridden

jspDestroy

- this is exceuted when ever Object is Destroyed
- It can be overridden

methods which contains "__" [underScrol] which are NOT OVERRIDDEN

2. JSP Scriptlets

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 three types of scripting elements:

1. **scriptlet tag**
2. **expression tag**
3. **declaration tag**

1. Scriptlet tag

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

```
<% java source code %>
```

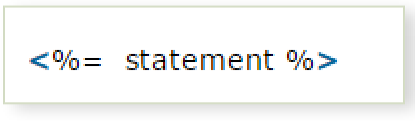
```
<html>
<body>
      <%
            out.print("welcome to jsp");
      %>
</body>
</html>
```

Output: `welcome to jsp`

It is placed in **_JspService ()** method. So method declarations not possible

2. Expression tag

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



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

```
<html>
<body>
      Date:<%=java.util.Calendar.getInstance().getTime()%>
</body>
</html>
```

```
Date:Thu Sep 22 19:10:14 IST 2016
```

It is placed in **JspInit()** method.

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

3. Declaration tag

- The JSP declaration tag is used to **declare fields and methods.**
- Code written inside the jsp declaration tag is placed **outside the service()** method

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

Variable Declaration

```
<html>
<body>
      <%!int data = 50;%>
      <%="Value of the variable is:" + data%>
</body>
</html>
```

Method Declaration

```
<html>
<body>
<%!
      int cube(int n){
            return n*n*n;
      }
%>

<%="Cube of 3 is:" + cube(3)%>
</body>
</html>
```

3. JSP Implicit objects

There are 9 jsp implicit objects. These objects are created by the web container that are available to all the jsp pages.

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

```
Index.jsp
<%
out.print("1.welcome to jsp");

String name=request.getParameter("a");
out.print("<br> 2.Request :"+name);

response.sendRedirect("http://www.google.com");
out.print("3.Responce :  ");

String cfg=config.getInitParameter("config");
out.print("<br> 4.Config ="+cfg);

String cxt=application.getInitParameter("context");
out.print("<br> 5.Application ="+cfg);

session.setAttribute("user","Satya");
out.print("<br> 6.Session ="+session.getAttribute("user"));
%>
```

```
<web-app>
        <servlet>
                <servlet-name>jsp</servlet-name>
                <jsp-file>/index.jsp</jsp-file>
                <init-param>
                        <param-name>config</param-name>
                        <param-value>iam Config Value</param-value>
                </init-param>
        </servlet>

        <servlet-mapping>
                <servlet-name>jsp</servlet-name>
                <url-pattern>/jsp</url-pattern>
        </servlet-mapping>

        <context-param>
                <param-name>context</param-name>
                <param-value>iam Context Value</param-value>
        </context-param>

</web-app>
```

7) pageContext implicit object

The pageContext object can be used to **set, get or remove attributes** from one of the following scopes:

- **Page** → PageContext.PAGE_SCOPE
- **Request** → PageContext.REQUEST_SCOPE
- **Session** → PageContext.SESSION_SCOPE
- **Application** → PageContext.APPLICATION_SCOPE

pageContext.setAttribute("name",” value”,PageContext.SESSION_SCOPE);

first.jsp

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);

<a href="second.jsp">second jsp page</a>

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

Second.jsp

```
<html>
<body>
<%
        String name=(String)pageContext.getAttribute("user",PageContext.SESSION_SCOPE);
        out.print("Hello "+name);
%>
</body>
</html>
```

8. Page implicit object

Page is an implicit object of type **Object class**

9. Exception

- Exception is an implicit object of type java.lang.Throwable class.
- This object can be used to print the exception.
- it can only be used in error pages

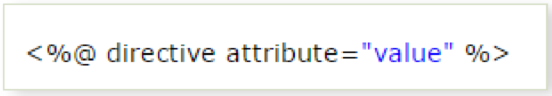
```
<%@ page isErrorPage="true" %>
<html>
<body>

Sorry following exception occured:<%= exception %>

</body>
</html>
```

4. 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:

1. **page directive**
2. **include directive**
3. **taglib directive**



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

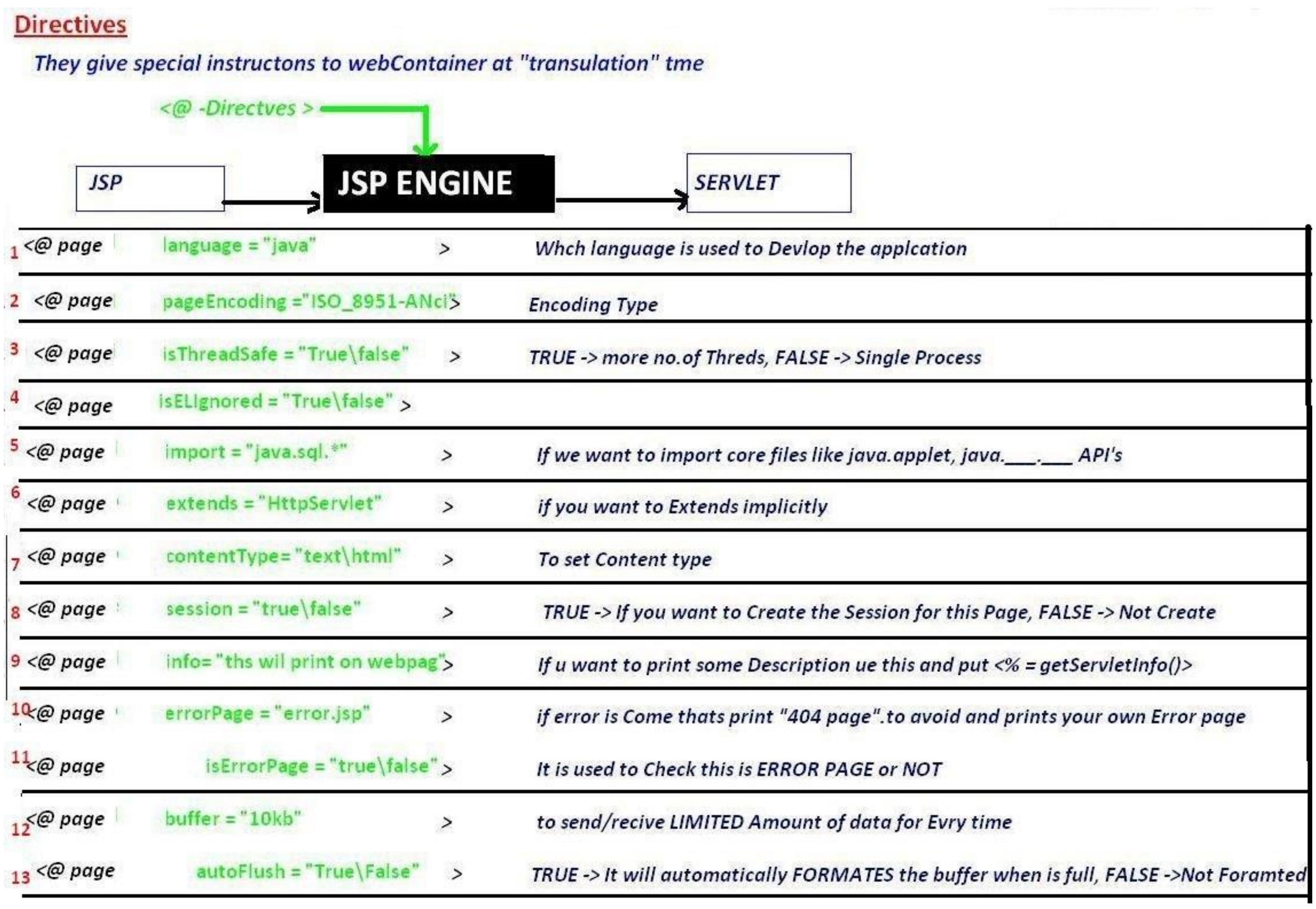
1. Page Directive

```
language        = java
pageEncoding ="ISO/ANCII"
isThreadSafe   = "True"
isELingnored   = "True"
```

```
import java.sql.*

class A extends B
{
   ContextType(text/html)

   Session

    info
}
```

```
<% @ errorPage="err.jsp"  isErrorPage("false") %>
```

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



Directives

They give special instructons to webContainer at "transulation" tme

1	<@ page	language = "java" >	Whch language is used to Devlop the applcation
2	<@ page	pageEncoding ="ISO_8951-ANci">	Encoding Type
3	<@ page	isThreadSafe = "True\false" >	TRUE -> more no.of Threds, FALSE -> Single Process
4	<@ page	isELIgnored = "True\false" >	
5	<@ page	import = "java.sql.*" >	If we want to import core files like java.applet, java.___.___ API's
6	<@ page	extends = "HttpServlet" >	if you want to Extends implicitly
7	<@ page	contentType= "text\html" >	To set Content type
8	<@ page	session = "true\false" >	TRUE -> If you want to Create the Session for this Page, FALSE -> Not Create
9	<@ page	info= "ths wil print on webpag">	If u want to print some Description ue this and put <% = getServletInfo()>
10	<@ page	errorPage = "error.jsp" >	if error is Come thats print "404 page".to avoid and prints your own Error page
11	<@ page	isErrorPage = "true\false" >	It is used to Check this is ERROR PAGE or NOT
12	<@ page	buffer = "10kb" >	to send/recive LIMITED Amount of data for Evry time
13	<@ page	autoFlush = "True\False" >	TRUE -> It will automatically FORMATES the buffer when is full, FALSE ->Not Foramted

```
<%@ page language="java" %>
<%@ page pageEncoding="ISO-8859-1"%>
<%@ page isELIgnored="false"%>
<%@ page isThreadSafe="true"%>
<%@ page errorPage="err.jsp" isErrorPage="false"%>
<%@ page import="java.lang.*"%>
<%@ page extends="java.lang.Object"%>
<%@ page contentType="text/html"%>
<%@ page session="true"%>
<%@ page info="Some Info Print on web page"%>
<%@ page buffer="8kb"%>
<%@ page autoFlush="true"%>
```

2. 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).

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

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

```
<html>
<body>

<%@ include file="header.html" %>

Today is: <%= java.util.Calendar.getInstance().getTime() %>

</body>
</html>
```

The include directive includes the original content, so the actual page size grows at runtime.

3. TagLib directive

- The JSP taglib directive is used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define the tags.
- We can insert custom tags by using this.

```
<%@ taglib uri="urioftheta glibrary" prefix="prefixoftaglibrary" %>
```

```
<html>
<body>
<%@ taglib uri="http://www.smlcodes.com/tags" prefix="mytag" %>
<mytag:currentDate/>
</body>
</html>
```

5. JSP Action Tags

The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are given below.

JSP Action Tags	Description
jsp:forward	Forwards the request and response to another resource.
jsp:include	Includes another resource.
jsp:param	Sets the parameter value. It is used in forward and include mostly.
jsp:useBean	Creates or locates bean object.
jsp:setProperty	Sets the value of property in bean object.
jsp:getProperty	Prints the value of property of the bean.
jsp:plugin	Embeds another components such as applet.
jsp:fallback	Can be used to print the message if plugin is working. It is used in jsp: plugin.

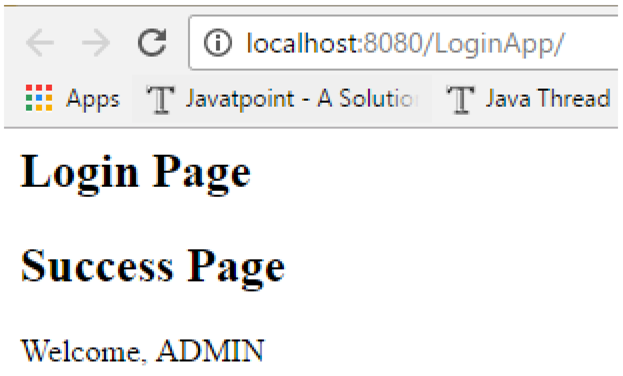
Forward, include, param example

Login.jsp

```
<h2>Login Page</h2>

<jsp:include page="Success.jsp">
      <jsp:param name="uname" value="ADMIN" />
      <jsp:param name="pwd" value="admin" />
</jsp:include>
```

Success.jsp

```
<h2>Success Page</h2>
 Welcome, <%= request.getParameter("uname")  %>
```



Login Page

Success Page

Welcome, ADMIN

Similarly we can use for <jsp:forward> also

UserBean Example

1. First we have choose the Input values for the Login page

```
<form action="set.jsp" method="post">
      Email <input type="text" name="email"><br>
      Pass  <input type="text" name="pwd"><br>
      <input type="submit" value="Login"><br>
</form>
```

2. We have to create UserBean class as per Input page parameters (email, pwd)

```
public class UserBean {
        String email;
        String pwd;
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public String getPwd() {
                return pwd;
        }
        public void setPwd(String pwd) {
                this.pwd = pwd;
        }
}
```

3. UserBean will set the values automatically by comparing property names

```
<jsp:useBean id="user" class="demo.UserBean">
   <jsp:setProperty name="user" property="email"/>
    <jsp:setProperty name="user" property="pwd"/>
</jsp:useBean>

<h3>getProperty Details</h3>
<jsp:getProperty name="user" property="email" /><br>
<jsp:getProperty name="user" property="pwd" /><br>
```

Here **name** is Object of bean class. & **propery** is the userbean property names

Output

```
getProperty Details
satyajohnny1@gmail.com
qw
```

jsp:plugin, jsp:fallbacks

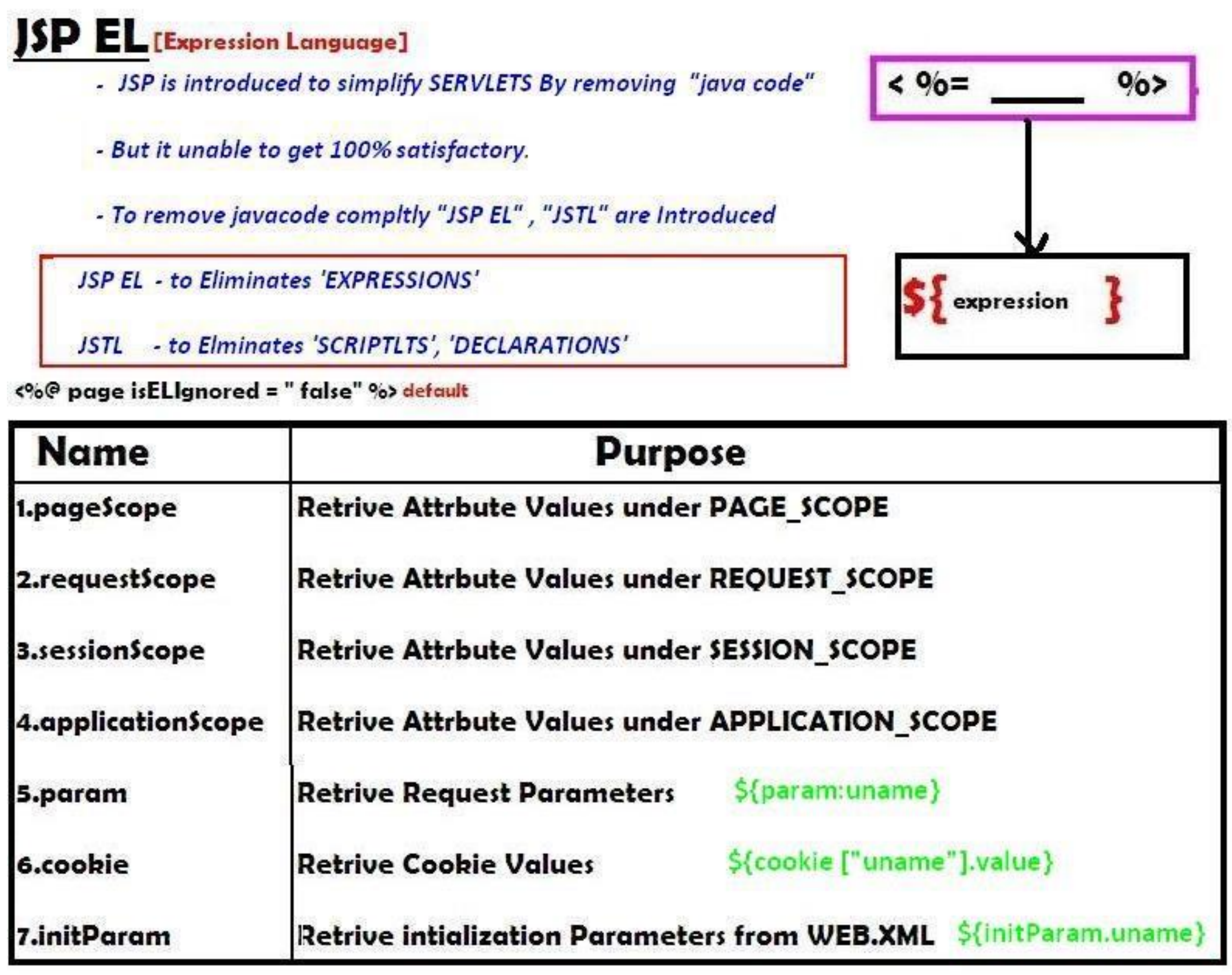
The **jsp:plugin** action tag is used to embed applet in the jsp file.

```
<jsp:plugin  height="500" width="500"  type="applet"  code="MouseDrag.class"  />
```

jsp:fallback action tag is used to display some message if Applet is not loading

```
<jsp:fallback>
        <p> Unable to start plugin </p>
</jsp:fallback>
```

6. JSP EL (JSP Expression Language)


JSP EL [Expression Language]

- *JSP is introduced to simplify SERVLETS By removing "java code"*
- *But it unable to get 100% satisfactory.*
- *To remove javacode compltly "JSP EL", "JSTL" are Introduced*

JSP EL - to Eliminates 'EXPRESSIONS'

JSTL - to Elminates 'SCRIPTLTS', 'DECLARATIONS'

<%= ____ %>

${ expression }

<%@ page isELIgnored = " false" %> default

Name	Purpose
1.pageScope	Retrive Attrbute Values under PAGE_SCOPE
2.requestScope	Retrive Attrbute Values under REQUEST_SCOPE
3.sessionScope	Retrive Attrbute Values under SESSION_SCOPE
4.applicationScope	Retrive Attrbute Values under APPLICATION_SCOPE
5.param	Retrive Request Parameters ${param:uname}
6.cookie	Retrive Cookie Values ${cookie ["uname"].value}
7.initParam	Retrive intialization Parameters from WEB.XML ${initParam.uname}

It simplifies for retrieving following types values mainly

1. **Request Paramters** → **req.getParamter("")**
2. **Init Patamter values** → **getInitParameter("")**
3. **Attribute Values** → **getAttribute("") in 4 scopes**
4. **Cookie values** → **getCookie("")**

```
<form action="jspel.jsp" method="post">
        Name <input type="text" name="name"><br>
        <input type="submit" value="Login"><br>
        <%
        session.setAttribute("pwd", "123456");
        %>
</form>
```

Jspel.jsp

```
Name :    ${ param.name }
Session: ${sessionScope.pwd}
```

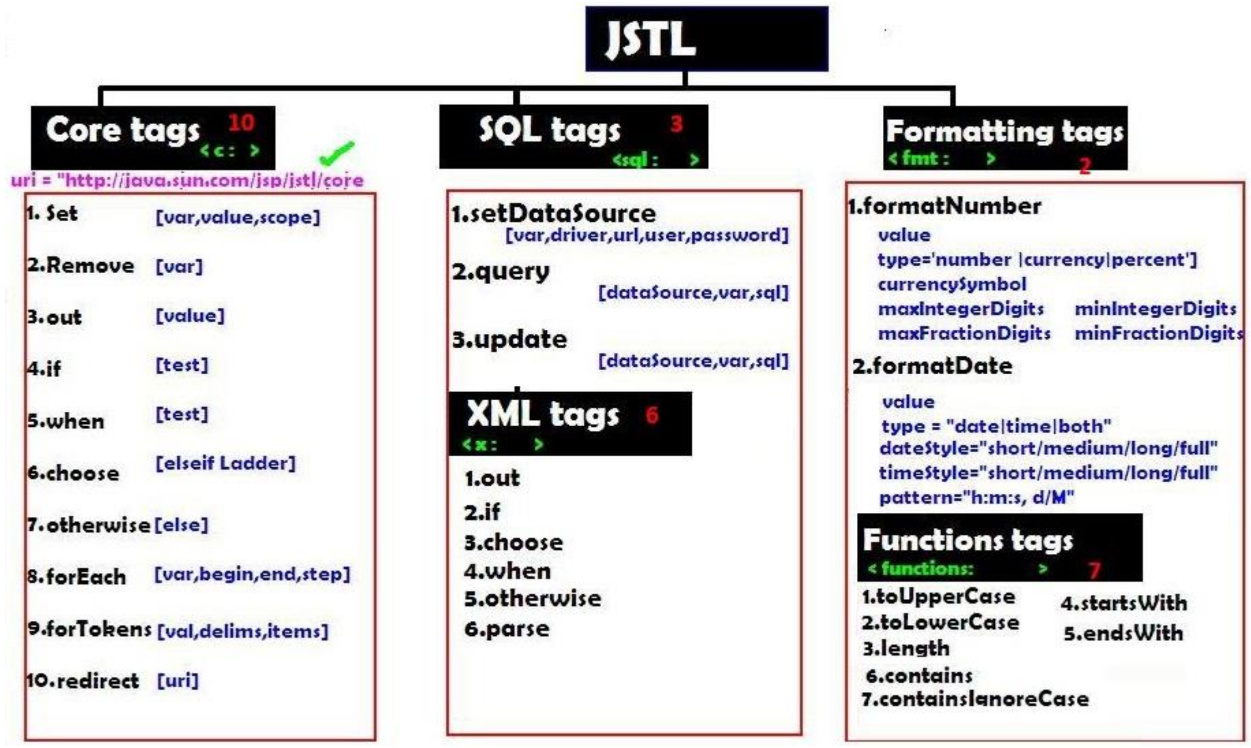
```
Name : Satya Session: 123456
```

7. JSP - Standard Tag Library (JSTL)

- JSP Standard Tag Library (JSTL) is a standard library of readymade tags.
- The JSTL contains several tags already implemented common functionalities.
- JSTL is external tags it is not come by default with JDK.we have to download [jstl.jar]() seperatly and placed in **lib/** folder

The JSTL tags can be classified, according to their functions, into following JSTL tag library groups that can be used when creating a JSP page:

1. Core Tags	**→ used for import,if, foreach loops**
2. Formatting tags	**→ used for formatting text, Date,number,URLencoding**
3. SQL tags	**→ Used for SQL operations like INSERT,SELECT., etc**
4. XML tags	**→ provides support for XML processing**
5. JSTL Functions	**→ provides support for string manipulation.**



1. core tags

we have to attach jstl/core url at the top of the jsp as below

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

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="salary" scope="session" value="${2000*2}"/>
Salary : <c:out value="${salary}"/>

<c:if test="${salary > 2000}">
   <p>My salary is: <c:out value="${salary}"/><p>
</c:if>
```

```
<!-- it is like if-else loop -->
<c:choose>
    <c:when test="${salary <= 0}">
       Salary is very low to survive.
    </c:when>
    <c:when test="${salary > 1000}">
        Salary is very good.
    </c:when>
    <c:otherwise>
        No comment sir...
    </c:otherwise>
</c:choose>

<!-- it is mainly used on Number prints[1,2,3,4,5]-->
<c:forEach var="i" begin="1" end="5">
   <c:out value="${i}"/><p>
</c:forEach>

<!-- it is mainly used on String-->
<c:forTokens items="Zara,nuha,roshy" delims="," var="name">
   <c:out value="${name}"/><p>
</c:forTokens>

<c:remove var="salary"/>
<br>Salary : <c:out value="${salary}"/>
```

2. SQL Tags

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


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


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


<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>
```

3. Formatting Tags

```
<h3> Format Number:</h3>
<c:set var="balance" value="120000.2309" />
<p>Cuurency <fmt:formatNumber value="${balance}" type="currency"/></p>
<p>Integr<fmt:formatNumber type="number" maxIntegerDigits="3" value="${balance}" />


<h3> Format Date:</h3>
<c:set var="now" value="<%=new java.util.Date()%>" />
<p>Only Time <fmt:formatDate type="time"  value="${now}" /></p>
<p>Only Date <fmt:formatDate type="date"  value="${now}" /></p>
<p>Time+Date <fmt:formatDate type="both"  value="${now}" /></p>
```

4. Function Tags (String Operations)

```
<c:set var="str" value="I am a test String"/>

<c:set var="lowStr" value="${fn:toLowerCase(string1)}" />

<c:set var="uprStr" value="${fn:toUpperCase(string1)}" />

<p>Length: ${fn:length(str)}</p>
```

8. JSP Custom Tags

For creating any custom tag, we need to follow following steps:

1. **Create the Tag handler class** (.java)
2. **Create the Tag Library Descriptor (TLD) file** and define tags**(.tld)**
3. **Create the JSP file that uses the Custom tags (.JSP)**

1.Create the Tag handler class (.java)

- To create the Tag Handler, we are inheriting the **TagSupport class**
- And override **doStartTag().**
- To write data for the jsp, we need to use the **JspWriter** class.its like res.getWriter()
- **PageContext** class provides **getOut()** method that returns **JspWriter** instance
- These classes are not Default with servlet-api.we have to download **jsp-api.jar**

```
public class MyTag extends TagSupport{

	public int doStartTag() throws JspException
	{
	   JspWriter out=pageContext.getOut();//returns the instance of JspWriter
	   out.print(Calendar.getInstance().getTime());//printing date using JspWriter
	    return SKIP_BODY;//will not evaluate the body content of the tag
	}
}
```

2. Create the Tag Library Descriptor (TLD) file and define tags(.tld)

Tag Library Descriptor (TLD) file contains information of tag and Tag Hander classes. It must be contained inside the **WEB-INF** directory.

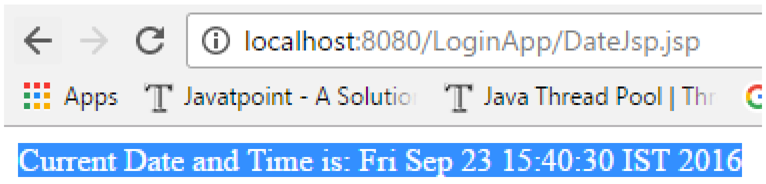
mytag.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
        PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
    "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">

<taglib>
      <tlib-version>1.0</tlib-version>
      <jsp-version>1.2</jsp-version>
      <short-name>simple</short-name>
      <uri>http://tomcat.apache.org/example-taglib</uri>
      <tag>
            <name>today</name>
            <tag-class>demo.MyTag</tag-class>
      </tag>
</taglib>
```

3. Create the JSP file that uses the Custom tags (.JSP)

DateJsp.jsp

```
<%@ taglib uri="WEB-INF/mytag.tld" prefix="m" %>
Current Date and Time is: <m:today/>
```



```
Current Date and Time is: Fri Sep 23 15:40:30 IST 2016
```