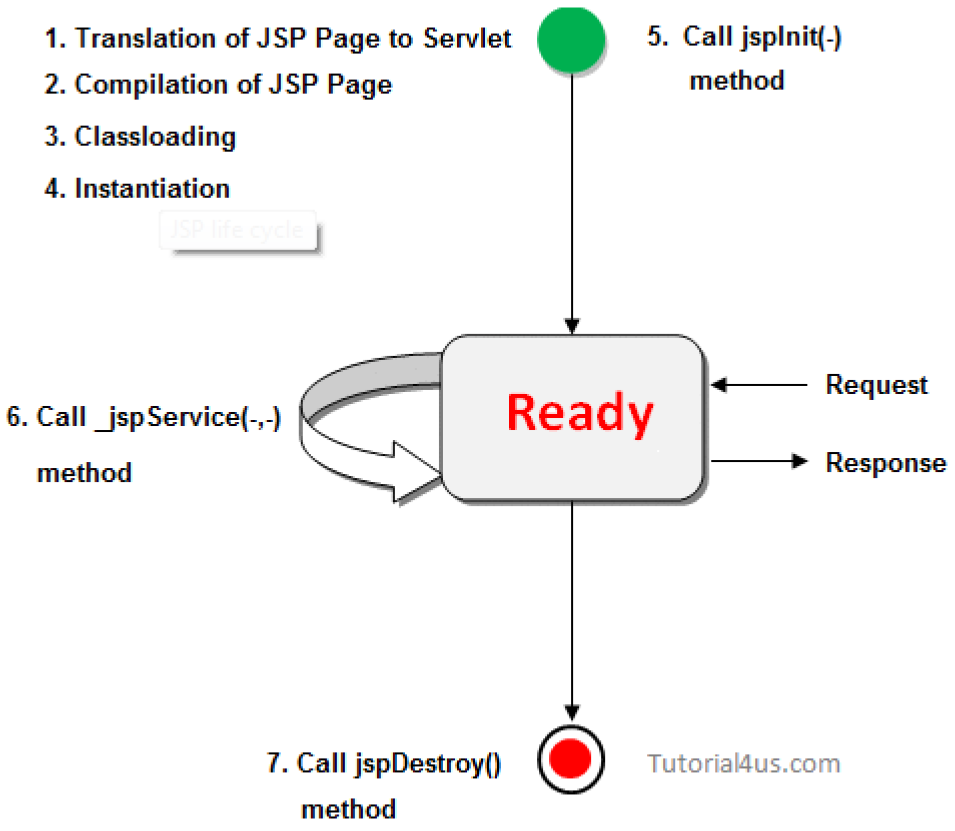
Introduction

JSP(Java Server Page) is an extension of Servlet which removes its drawbacks. In web application instead of generating html from the program, the program is embedded into the HTML using JSP.

JSP Life Cycle

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 translate a JSP into Servlet.


1. Translation of JSP Page to Servlet
2. Compilation of JSP Page
3. Classloading
4. Instantiation
5. Call jspInit(-) method
Ready
Request
Response
6. Call _jspService(-,-) method
7. Call jspDestroy() method
Tutorial4us.com

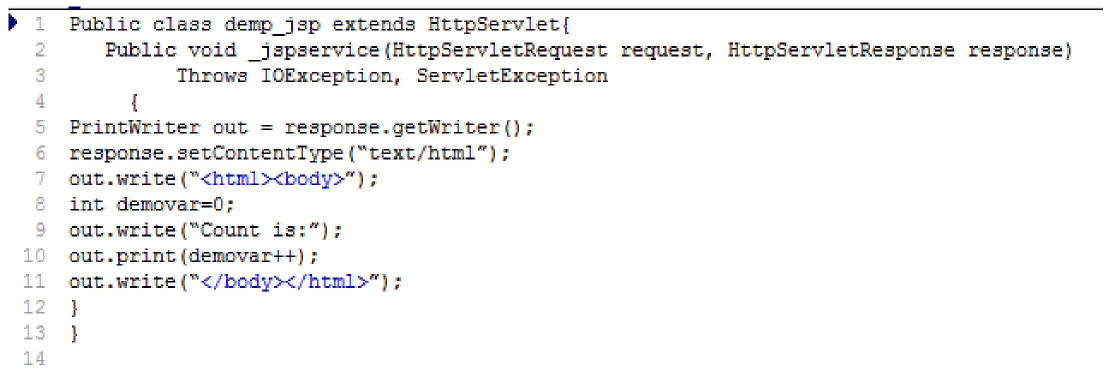
1. Translation of the JSP Page

Demo.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Demo JSP</title>
</head>
<body>
<%!
int demovar=0;%>
Count is:
<% out.println(++demovar); %>
</body>
```

```
</html>
```

JSP converted to demo_jsp.java Servlet



```
Public class demp_jsp extends HttpServlet{
   Public void _jspservice(HttpServletRequest request, HttpServletResponse response)
         Throws IOException, ServletException
     {
PrintWriter out = response.getWriter();
response.setContentType("text/html");
out.write("<html><body>");
int demovar=0;
out.write("Count is:");
out.print(demovar++);
out.write("</body></html>");
}
}
```

2. **Compilation of the JSP Page**
   - ✓ The generated java servlet file is compiled into java servlet class
   - ✓ The translation of java source page to its implementation class can happen at any time between the deployment of JSP page into the container and processing of the JSP page.
   - ✓ In the above pictorial description demo_jsp.java is compiled to a class file demo_jsp.class
3. **Classloading**

   - ✓ Servlet class that has been loaded from JSP source is now loaded into the container

4. **Instantiation**

   - ✓ In this step the object i.e. the instance of the class is generated.
   - ✓ The container manages one or more instances of this class in the response to requests and other events. Typically, a JSP container is built using a servlet container. A JSP container is an extension of servlet container as both the container support JSP and servlet.
   - ✓ A JSPPage interface which is provided by container provides init() and destroy () methods.
   - ✓ There is an interface HttpJSPPage which serves HTTP requests, and it also contains the service method.

5. **Initialization**

public void jspInit(){

//initializing the code

}

   - ✓ _jspinit() method will initiate the servlet instance which was generated from JSP and will be invoked by the container in this phase.
   - ✓ Once the instance gets created, init method will be invoked immediately after that
   - ✓ It is only called once during a JSP life cycle, the method for initialization is declared as shown above

6. **Request processing**

void _jspservice(HttpServletRequest request HttpServletResponse response){

```
        //handling all request and responses

}
```

- ✓ _jspservice() method is invoked by the container for all the requests raised by the JSP page during its life cycle
- ✓ For this phase, it has to go through all the above phases and then only service method can be invoked.
- ✓ It passes request and response objects
- ✓ This method cannot be overridden
- ✓ The method is shown above: It is responsible for generating of all HTTP methods i.e GET, POST, etc.

7. Destroy

```
public void _jspdestroy(){

//all clean up code

}
```

- ✓ _jspdestroy() method is also invoked by the container
- ✓ This method is called when container decides it no longer needs the servlet instance to service requests.
- ✓ When the call to destroy method is made then, the servlet is ready for a garbage collection
- ✓ This is the end of the life cycle.
- ✓ We can override jspdestroy() method when we perform any cleanup such as releasing database connections or closing open files.

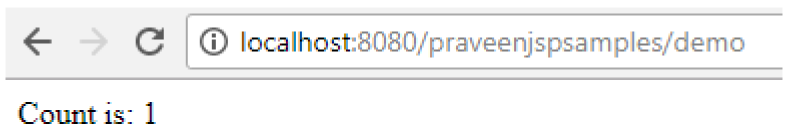
Running JSP file in Tomcat webserver

Place the above demo.jsp in WebContent folder and in web.xml(present in WEB-INF) make the below changes

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
      id="WebApp_ID" version="3.1">
      <display-name>praveenjspsamples</display-name>
      <servlet>
            <servlet-name>DemoJSP</servlet-name>
            <jsp-file>/demo.jsp</jsp-file>
      </servlet>
      <servlet-mapping>
            <servlet-name>DemoJSP</servlet-name>
            <url-pattern>/demo</url-pattern>
      </servlet-mapping>

</web-app>
```

http://localhost:8080/praveenjspsamples/demo



Count is: 1

Difference Between Servlet and JSP

	Servlet	JSP
1	Servlet is faster than jsp	JSP is slower than Servlet because it first translate into java code then compile.
2	In Servlet, if we modify the code then we need recompilation, reloading, restarting the server> It means it is time consuming process.	In JSP, if we do any modifications then just we need to click on refresh button and recompilation, reloading, restart the server is not required.
3	Servlet is a java code.	JSP is tag based approach.
4	In Servlet, there is no such method for running JavaScript at client side.	In JSP, we can use the client side validations using running the JavaScript at client side.
5	To run a Servlet you have to make an entry of Servlet mapping into the deployment descriptor file i.e. web.xml file externally.	For running a JSP there is no need to make an entry of Servlet mapping into the web.xml file externally, you may or not make an entry for JSP file as welcome file list.
6	Coding of Servlet is harden than jsp.	Coding of jsp is easier than Servlet because it is tag based.
7	In MVC pattern, Servlet plays a controller role.	In MVC pattern, JSP is used for showing output data i.e. in MVC it is a view.
8	Servlet accept all protocol request.	JSP will accept only http protocol request.
9	In Servlet, service() method need to override.	In JSP no need to override service() method.
10	In Servlet, by default session management is not enabled we need to enable explicitly.	In JSP, session management is automatically enabled.
11	In Servlet we do not have implicit object. It means if we want to use an object then we need to get object explicitly form the servlet.	In JSP, we have implicit object support.
12	In Servlet, we need to implement business logic, presentation logic combined.	In JSP, we can separate the business logic from the presentation logic by uses javaBean technology.
13	In Servlet, all package must be imported on top of the servlet.	In JSP, package imported anywhere top, middle and bottom.

Scripting Element

In JSP, java code can be written inside the jsp page using the scriptlet tag. JSP Scripting element are written inside <% %> tags. These code inside <% %> tags are processed by the JSP engine during translation of the JSP page.

Jsp scripting elements are classified into two types are;

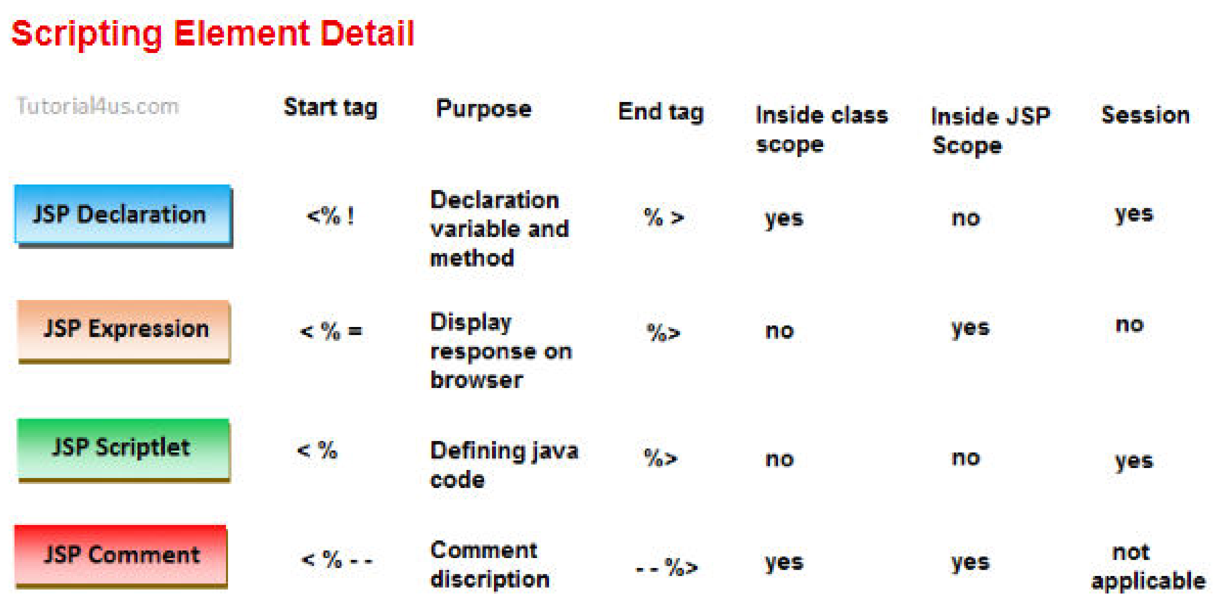
- ✓ Language Based Scripting Element
- ✓ Advance Scripting Elements (Expression Language)

Language Based Scripting Element

These are used to defined script in jsp page, this is traditional approach to define the script in jsp page.

Language Based Scripting Element are classified into 4 types, they are;

- ✓ Comment Tag
- ✓ Declaration Tag
- ✓ Expression Tag
- ✓ Scriptlet Tag


Scripting Element Detail

Tutorial4us.com	Start tag	Purpose	End tag	Inside class scope	Inside JSP Scope	Session
JSP Declaration	<%!	Declaration variable and method	% >	yes	no	yes
JSP Expression	< % =	Display response on browser	%>	no	yes	no
JSP Scriptlet	< %	Defining java code	%>	no	no	yes
JSP Comment	< % - -	Comment discription	- - %>	yes	yes	not applicable

1. JSP Declaration Tag

This is used to declare variable and methods in jsp page will be translate and define as class scope declaration in .java file.

The variable and methods will become global to that jsp. It means in that jsp we can use those variables anywhere and we can call those method anywhere.

At the time of translation container inserts the declaration tag into the class. So the variables become instants variables and methods will become instants methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto-generated Servlet, so it does not get memory at each request.

Syntax: <%! variable declaration or method declaration %>

2. JSP Expression Tag

Expression tag is used, to find the result of the given expression and send that result back to the client browser. In other words; These are used to show or express the result or response on browser. We can use this as a display result of variable and invoking method.

Each Expression tag of jsp will be internally converted into out.print() statement. In jsp out is an implicit object.

The expression tags are converted into out.print() statement and insert into the _jspService(-,-) of the servlet class by the container.

The expression tags written in a jsp are executed for each request, because _jspService(-,-) is called for each request.

One expression tag should contains one java expression only. In a jsp we can use expression tag for any number of times.

Syntax: <%= expression %>

Expression does not need any semicolon (;) by default it places under jsp service() method scope.

Note: In jsp, the implicit object are allowed into the tags, which are inserted into _jspService(-,-). An expression tag goes to _jspService(-,-) only. So implicit object are allowed in the expression tag.

3. JSP Scriptlet Tag

It use to define or insert java code in a jsp. Every java statement should followed by semicolon (;). It will be place into jspService() method.

The scriptlet tags goes to _jspService(-,-), during translation. It means scriptlet tags are executed for each request. Because _jspService(-,-) is called for each request.

We can use implicit object of jsp in a scriptlet tag also because scriptlet tag goes to _jspService(-,-) only.

In a jsp implicit object are allowed in expression tags and scriptlet tags but not in the declaration tags.

Syntax: <% java code %>

Difference between the jsp scriptlet tag and jsp declaration tag?

Jsp Scriptlet Tag	Jsp Declaration Tag
The jsp 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.

JSP Implicit Objects

JSP provides standard or predefined implicit objects, which can use directly in JSP page using JSP Scriptlet. The implicit objects are Servlet API class type and created by JSP containers

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

The available implicit objects are out, request, response, page, pageContext, exception, config, session and application.

List of all 9 implicit object are;

	Class type	**Object**
1	HttpServletRequest	request
2	HttpServletResponse	response
3	ServletConfig	config
4	ServletContext	application
5	HttpSession	session
6	JspWriter/PrintWriter	out
7	Exception	exception
8	PageContext	pagecontext
9	Object	page

All implicit objects of jsp are accessible within the expression and scriptlet of the jsp, but not accessible in the declaration tags of the jsp.

In a jsp session and exception object are not available always. Because session object depends on session attribute of the page directive and exception depends on isErrorPage attribute of the page directive.

In a jsp, session object is available, if session equals to true in page directive.

In a jsp an exception object is available, if isErrorPage is equal to true in page directive.

Request Implicit Object

The JSP request is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

Example of request implicit object where we are printing the name of the user with welcome message.

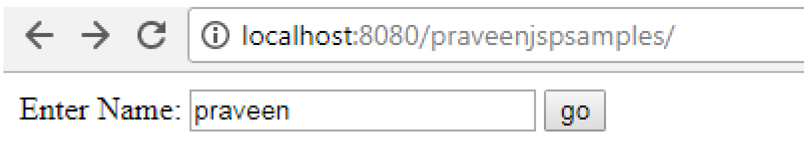
index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
```

```
<title>Welcome Page</title>
</head>
<body>
<form action="welcome.jsp">
Enter Name: <input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

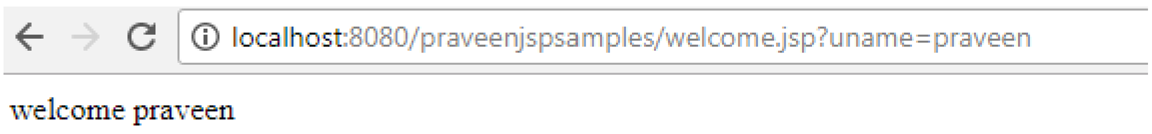
welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</body>
</html>
```



localhost:8080/praveenjspsamples/

Enter Name: praveen go



localhost:8080/praveenjspsamples/welcome.jsp?uname=praveen

welcome praveen

Response Implicit Object

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.

It can be used to add or manipulate response such as redirect response to another resource, send error etc.

Example of response implicit object where we are redirecting the response to the Google.

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Welcome Page</title>
```

```
</head>
<body>
<form action="welcome.jsp">
Enter Name: <input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

redirect.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Redirect Page</title>
</head>
<body>
<%
response.sendRedirect("http://www.google.com");
%>
</body>
</html>
```

Config Implicit Object

config object is an implicit object of type ServletConfig and it is created by the container, whenever servlet object is created. This object can be used to get initialization parameter for a particular JSP page.

config object is created by the web container for every jsp page. It means if a web application has three jsp pages then three config object are created.

In jsp, it is not mandatory to configure in web.xml. If we configure a jsp in web.xml then the logical name and init parameter given in web.xml file are stored into config object by the container.

config object is accessible, only if the request is given to the jsp by using its url pattern, but not with name of the jsp.

config object is accessible, only if the request is given to the jsp by using its url pattern, but not with name of the jsp.

Example

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
       id="WebApp_ID" version="3.1">
```

```
      <display-name>praveenjspsamples</display-name>
      <servlet>
            <servlet-name>WelcomeJSP</servlet-name>
            <jsp-file>/welcome.jsp</jsp-file>
            <init-param>
                  <param-name>driverName</param-name>
                  <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
            </init-param>
      </servlet>
      <servlet-mapping>
            <servlet-name>WelcomeJSP</servlet-name>
            <url-pattern>/welcome</url-pattern>
      </servlet-mapping>
</web-app>
```

welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
<%
String name=request.getParameter("uname");
out.print("Welcome "+name);
String driver=config.getInitParameter("driverName");
out.print("Driver name is="+driver);
%>

</body>
</html>
```

Page Implicit Object

In JSP, page is an implicit object of type Object class. When a jsp is translated to an internal Servlet, we can find the following statement in the service() method of servlet.

Object page=this;

For using this object it must be cast to Servlet type.

For example:

<% (HttpServlet)page.log("message"); %>

Since, it is of type Object it is less used because you can use this object directly in jsp.

For example:

<% this.log("message"); %>

Session Implicit Object

In JSP, session is an implicit object of type HttpSession.This object used to set,get or remove attribute or to get session information.

Example

welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
<%
String name=request.getParameter("uname");
out.print("Welcome "+name);
session.setAttribute("user",name);
%>
<a href="hello.jsp">Click Hello</a>
</body>
</html>
```

hello.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello Page</title>
</head>
<body>
<%
String name=(String)session.getAttribute("user");
out.print("Hello "+name);
%>
</body>
</html>
```

Exception Implicit Object

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

In this example we are taking two integer inputs from user and then we are performing division between them. We have used exception implicit object to handle any kind of exception in the below example.

division.html

```
<!DOCTYPE html>
<html>
```

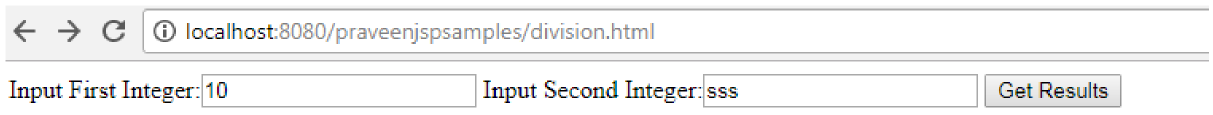
```
<head>
<meta charset="ISO-8859-1">
<title>Enter two Integers for Division</title>
</head>
<body>
<form action="division.jsp">
Input First Integer:<input type="text" name="firstnum" />
Input Second Integer:<input type="text" name="secondnum" />
<input type="submit" value="Get Results"/>
</form>
</body>
</html>
```

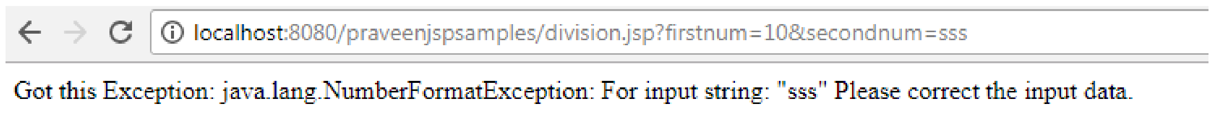
division.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Division Calculation Page</title>
</head>
<body>
<%@ page errorPage="exception.jsp" %>
<%
String num1=request.getParameter("firstnum");
String num2=request.getParameter("secondnum");
int v1= Integer.parseInt(num1);
int v2= Integer.parseInt(num2);
double res= (double)v1/v2;
out.print("Output is: "+ res);
%>
</body>
</html>
```

exception.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Exception Page</title>
</head>
<body>
<%@ page isErrorPage="true" %>
Got this Exception: <%= exception %>
Please correct the input data.
</body>
</html>
```


localhost:8080/praveenjspsamples/division.html


localhost:8080/praveenjspsamples/division.jsp?firstnum=10&secondnum=sss

Got this Exception: java.lang.NumberFormatException: For input string: "sss" Please correct the input data.

Application Implicit Object

In JSP, application is an implicit object of type ServletContext. this application object in the Servlet programming is ServletContext.

For all jsp in a web application, there must be a single application object with application object we can share the data from one JSP to any other JSP in the web application.

The instance of ServletContext is created only once by the web container when application or project is deployed on the server.

This object can be used to get initialization parameter from configuration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

For example

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
       id="WebApp_ID" version="3.1">
       <display-name>praveenjspsamples</display-name>
       <servlet>
              <servlet-name>WelcomeJSP</servlet-name>
              <jsp-file>/welcome.jsp</jsp-file>
              <init-param>
                     <param-name>driverName</param-name>
                     <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
              </init-param>
       </servlet>
       <servlet-mapping>
              <servlet-name>WelcomeJSP</servlet-name>
              <url-pattern>/welcome</url-pattern>
       </servlet-mapping>
       <context-param>
              <param-name>driverName</param-name>
              <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
       </context-param>
</web-app>
```

welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
<%
String name=request.getParameter("uname");
out.print("Welcome "+name);
session.setAttribute("user",name);
//String driver=config.getInitParameter("driverName");
//out.print("Driver name is="+driver);
String driver=application.getInitParameter("driverName");
out.print("driver name is="+driver);
%>
<a href="hello.jsp">Click Hello</a>
<%-- <% this.log("message"); %> --%>

</body>
</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.

For example

welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
<%
String name=request.getParameter("uname");
out.print("Welcome "+name);
//session.setAttribute("user",name);
//String driver=config.getInitParameter("driverName");
//out.print("Driver name is="+driver);
String driver=application.getInitParameter("driverName");
out.print("driver name is="+driver);

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

%>
<a href="hello.jsp">Click Hello</a>
```

```
<%-- <% this.log("message"); %> --%>

</body>
</html>
```

hello.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello Page</title>
</head>
<body>
<%
//String name=(String)session.getAttribute("user");
String
name=(String)pageContext.getAttribute("userPage",PageContext.SESSION_SCOPE);
out.print("Hello "+name);

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

JSP Directive Elements

The directive elements are used to do page related operation during page translation time. JSP supports 3 types of directive elements, they are;.

- ✓ Page directive
- ✓ Include directive
- ✓ Taglib directive

Syntax of directive elements

<@ directive attribute="value" %>

Page Directive

The page directive defines attributes that apply to an entire JSP page. This element is used to define default of explicit operation to the jsp.

Syntax

<@ page attribute="value" %>

Attributes of JSP page directive

- ✓ import
- ✓ contentType
- ✓ extends
- ✓ language
- ✓ buffer
- ✓ info

- ✓ isELIgnored
- ✓ isThreadSafe
- ✓ autoFlush
- ✓ session
- ✓ pageEncoding
- ✓ errorPage
- ✓ isErrorPage

JSP Include Directive

It is used to include some other pages into the JSP document, It may be JSP file, html file or text file. This is known as static include because the target page is located and included at the time of page compilation.

The jsp page is translated only once so it will be better to include static resource.

Advantage of Include directive

Code Re-usability

Syntax of Include directive elements

<%@ include file="resourceName"%>

For Example

<html>

<body>

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

</body>

</html>

Note: The include directive includes the original content, so the actual page size grows at run-time.

Taglib Directive

This is used to use custom tags in JSP page, here the custom tag may be user defined ot JSTL tag or struts, JSF,..... etc.

Syntax

<@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>

Attributes of taglib directive

- ✓ uri

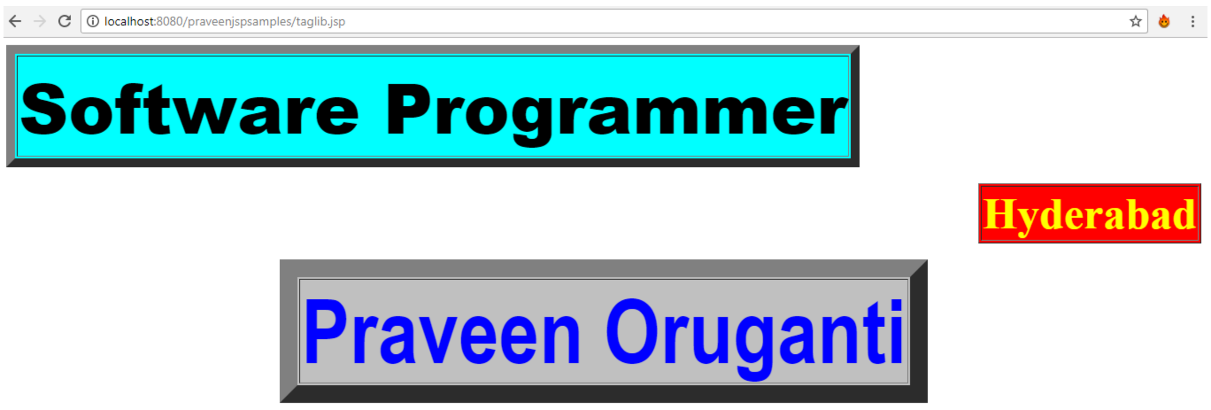
For example

heading2.tag

```
<%@ attribute name="align" required="true" %>
<%@ attribute name="bgColor" required="true" %>
<%@ attribute name="border" required="true" %>
<%@ attribute name="fgColor" required="true" %>
<%@ attribute name="font" required="true" %>
<%@ attribute name="size" required="true" %>
<TABLE ALIGN="${align}"
       BGCOLOR="${bgColor}"
       BORDER="${border}">
  <TR><TH>
      <SPAN STYLE="color: ${fgColor};
                   font-family: ${font};
                   font-size: ${size}px">
      <jsp:doBody/></SPAN>
</TABLE><BR CLEAR="ALL"><BR>
```

taglib.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Taglib Demo JSP</title>
</head>
<body>
<%@ taglib tagdir="/WEB-INF/tags" prefix="csajsp" %>
<csajsp:heading2 align="LEFT" bgColor="CYAN"
                 border="10" fgColor="BLACK"
                 font="Arial Black" size="78">
  Software Programmer
</csajsp:heading2>
<csajsp:heading2 align="RIGHT" bgColor="RED"
                 border="1" fgColor="YELLOW"
                 font="Times New Roman" size="50">
  Hyderabad
</csajsp:heading2>
<csajsp:heading2 align="CENTER" bgColor="#C0C0C0"
                 border="20" fgColor="BLUE"
                 font="Arial Narrow" size="100">
 Praveen Oruganti
</csajsp:heading2>
</body>
</html>
</body>
</html>
```


localhost:8080/praveenjspsamples/taglib.jsp

Software Programmer

Hyderabad

Praveen Oruganti

JSP – Standard Tag Library (JSTL)

JSTL stands for The JSP standard Tag Library is used to simplify the JSP development to represent a set of tags. It gives support to iteration and conditional tasks, underpins the tags to control the xml files, it underpins

skeleton to the merged existed custom tags. The following are the advantages of JSTL.

- It have many tags for fast development which simplifies the JSP.
- It was used for reusability of the code in various pages.
- For using JSTL, scriptlet tag can be avoided.

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:

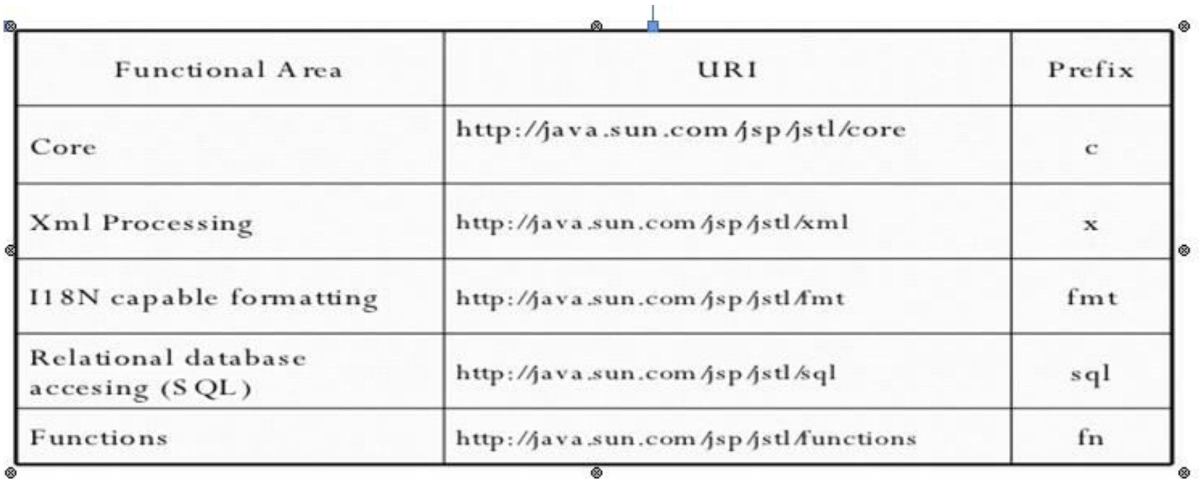
☐ Core Tags → used for import,if, foreach loops

☐ Formatting tags → used for formatting text, Date,number,URLencoding

☐ SQL tags → used for SQL operations like INSERT,SELECT., etc

☐ XML tags → provides support for XML processing

☐ JSTL Functions → provides support for string manipulation.



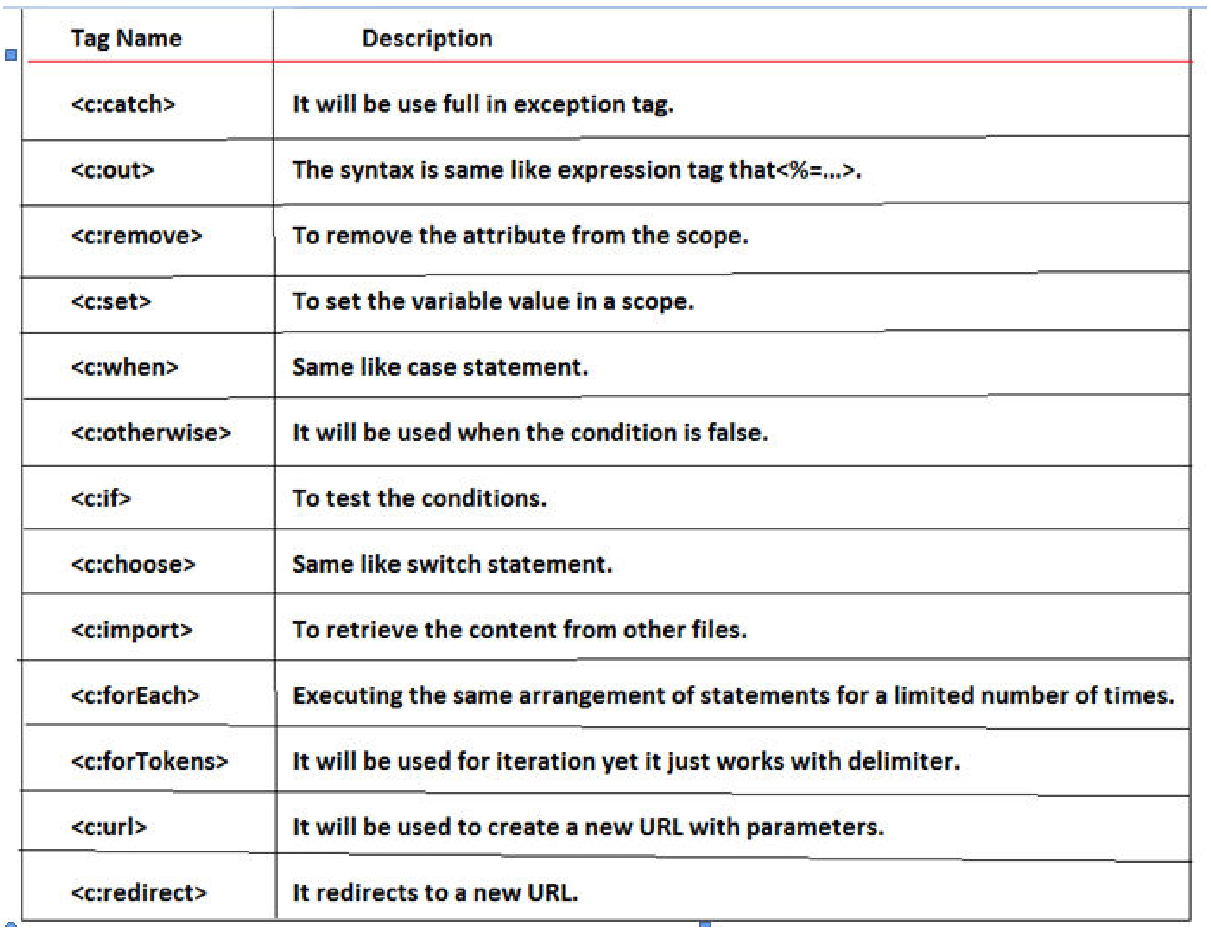
Functional Area	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
Xml Processing	http://java.sun.com/jsp/jstl/xml	x
I18N capable formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Relational database accesing (SQL)	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

Core tags

Core tags provide variable support, flow control and URL administration and so on

The following is the syntax to represent core tags.

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

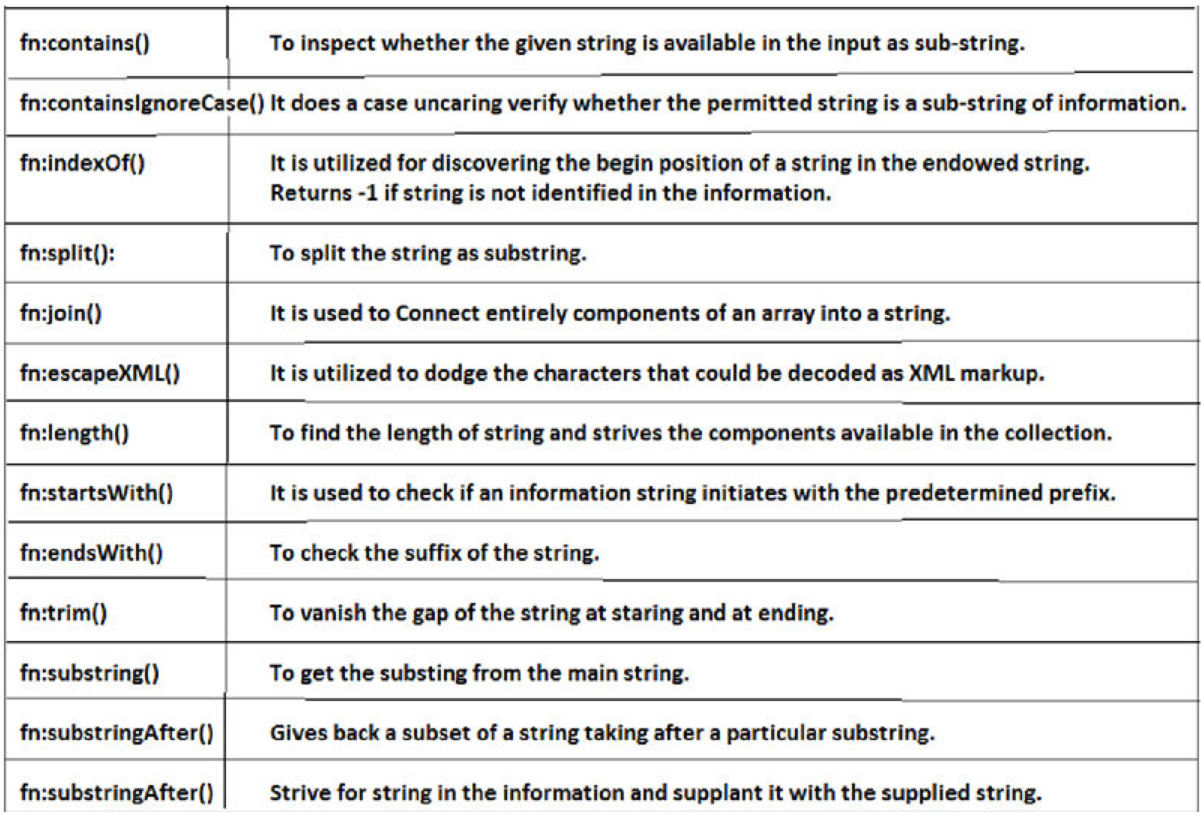


Tag Name	Description
<c:catch>	It will be use full in exception tag.
<c:out>	The syntax is same like expression tag that<%=...>.
<c:remove>	To remove the attribute from the scope.
<c:set>	To set the variable value in a scope.
<c:when>	Same like case statement.
<c:otherwise>	It will be used when the condition is false.
<c:if>	To test the conditions.
<c:choose>	Same like switch statement.
<c:import>	To retrieve the content from other files.
<c:forEach>	Executing the same arrangement of statements for a limited number of times.
<c:forTokens>	It will be used for iteration yet it just works with delimiter.
<c:url>	It will be used to create a new URL with parameters.
<c:redirect>	It redirects to a new URL.

Function Tags

Function tags gives support for sting manipulation.

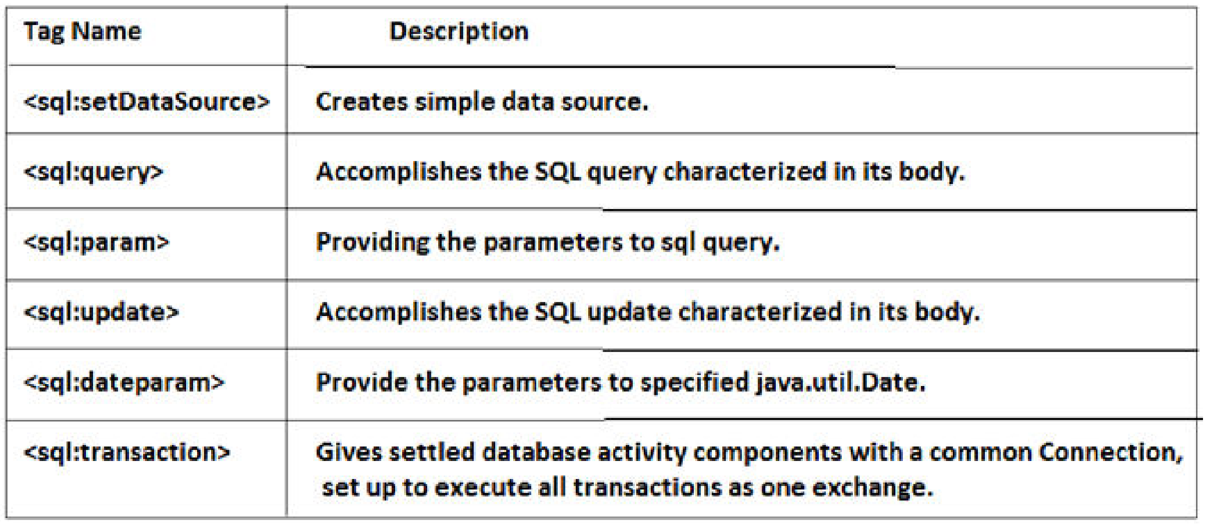
URL tag is http://java.sun.com/jsp/jstl/functions and where fn is the prefix.



fn:contains()	To inspect whether the given string is available in the input as sub-string.
fn:containsIgnoreCase()	It does a case uncaring verify whether the permitted string is a sub-string of information.
fn:indexOf()	It is utilized for discovering the begin position of a string in the endowed string. Returns -1 if string is not identified in the information.
fn:split():	To split the string as substring.
fn:join()	It is used to Connect entirely components of an array into a string.
fn:escapeXML()	It is utilized to dodge the characters that could be decoded as XML markup.
fn:length()	To find the length of string and strives the components available in the collection.
fn:startsWith()	It is used to check if an information string initiates with the predetermined prefix.
fn:endsWith()	To check the suffix of the string.
fn:trim()	To vanish the gap of the string at staring and at ending.
fn:substring()	To get the substing from the main string.
fn:substringAfter()	Gives back a subset of a string taking after a particular substring.
fn:substringAfter()	Strive for string in the information and supplant it with the supplied string.

JSTL SQL Tags

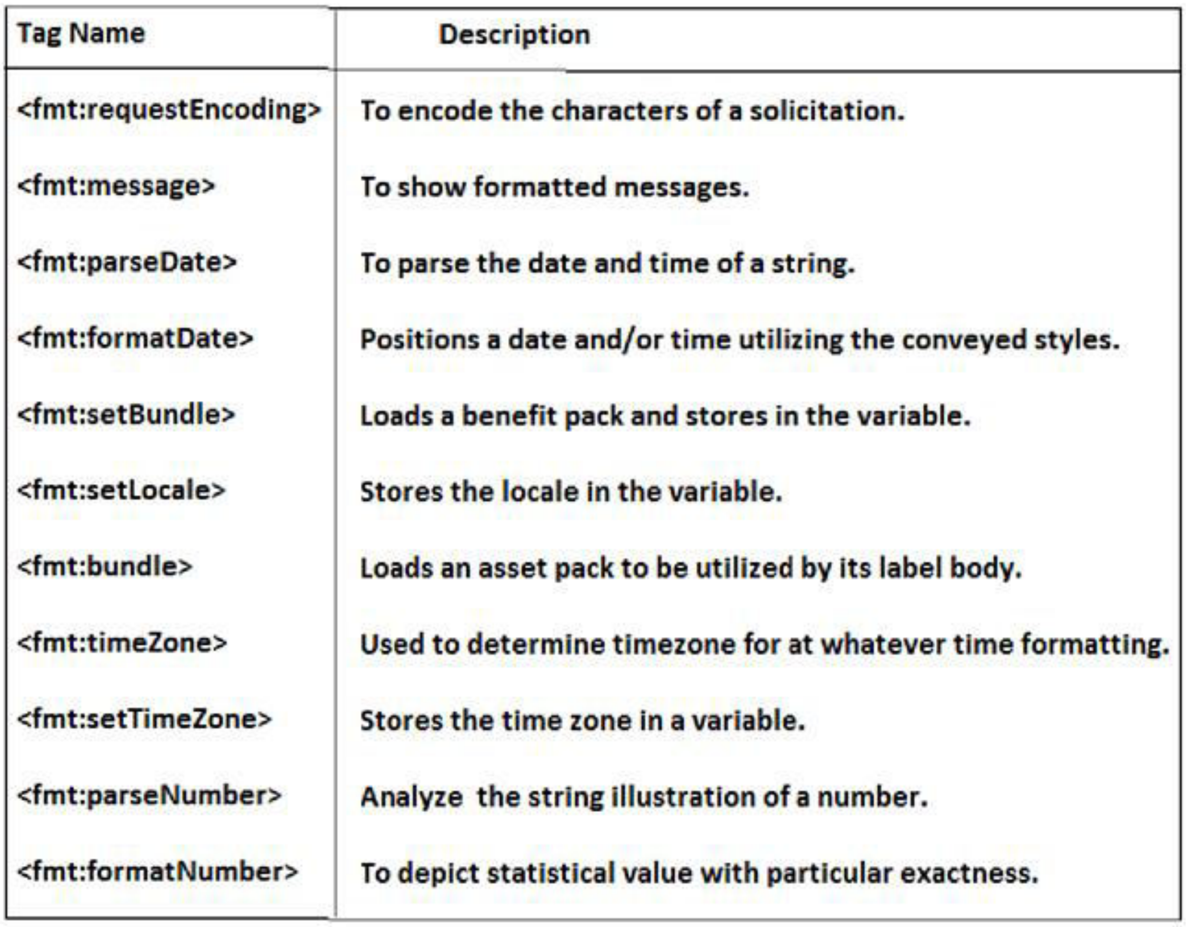
Provides SQL support and the URL for SQL tag is http://java.sun.com/jsp/jstl/sql and where prefix is sql.



Tag Name	Description
<sql:setDataSource>	Creates simple data source.
<sql:query>	Accomplishes the SQL query characterized in its body.
<sql:param>	Providing the parameters to sql query.
<sql:update>	Accomplishes the SQL update characterized in its body.
<sql:dateparam>	Provide the parameters to specified java.util.Date.
<sql:transaction>	Gives settled database activity components with a common Connection, set up to execute all transactions as one exchange.

JSTL Internationalization Tags / Formatting tags

The tag gives backing to message formating, number and date designing and so on. What's more, the url's tag is http://java.sun.com/jsp/jstl/fmt and where fmt is the prefix.



Tag Name	Description
<fmt:requestEncoding>	To encode the characters of a solicitation.
<fmt:message>	To show formatted messages.
<fmt:parseDate>	To parse the date and time of a string.
<fmt:formatDate>	Positions a date and/or time utilizing the conveyed styles.
<fmt:setBundle>	Loads a benefit pack and stores in the variable.
<fmt:setLocale>	Stores the locale in the variable.
<fmt:bundle>	Loads an asset pack to be utilized by its label body.
<fmt:timeZone>	Used to determine timezone for at whatever time formatting.
<fmt:setTimeZone>	Stores the time zone in a variable.
<fmt:parseNumber>	Analyze the string illustration of a number.
<fmt:formatNumber>	To depict statistical value with particular exactness.

Custom Tags in JSP

When EL and Standard Action elements aren't enough to remove scriptlet code from your JSP Page, you can use Custom Tags. Custom tags are nothing but user-defined tags.

Custom tags are an excellent way to abstract the complexity of business logic from the presentation of Web pages in a way that is easy for the Web author to use and control. It also allows for reusability as custom tags can be used again and again.

1) Tag handler class: In this class we specify what our custom tag will do when it is used in a JSP page.

2) TLD file: Tag descriptor file where we will specify our tag name, tag handler class and tag attributes.

3) JSP page: A JSP page where we will be using our custom tag.

Tag Handler Class

We can create a Tag Handler class in two different ways:

By implementing one of three interfaces : SimpleTag, Tag or BodyTag, which define methods that are invoked during the life cycle of the tag.

By extending an abstract base class that implements the SimpleTag, Tag, or BodyTag interfaces. The SimpleTagSupport, TagSupport, and BodyTagSupport classes implement the SimpleTag, Tag andBodyTag interfaces . Extending these classes relieves the tag handler class from having to implement all methods in the interfaces and also provides other convenient functionality.

Tag Library Descriptor

A Tag Library Descriptor is an XML document that contains information about a library as a whole and about each tag contained in the library. TLDs are used by the web container to validate the tags and also by JSP page development tools.

Tag library descriptor file must have the extension .tld and must be packaged in the /WEB-INF/ directory or subdirectory of the WAR file or in the /META-INF/ directory or subdirectory of a tag library packaged in a JAR.

Example of Custom Tag

In our example, we will be creating a Tag Handler class that extends the TagSupport class. When we extend this class, we have to override the method doStartTag(). There are two other methods of this class namely doEndTag() and release(), that we can decide to override or not depending on our requirement.

--CountMatches.java---

```
package com.ashoksoft.apps;



import java.io.IOException;

import javax.servlet.jsp.*;

import org.apache.commons.lang.StringUtils;



public class CountMatches extends TagSupport {

private String inputstring;

private String lookupstring;



public String getInputstring() {

return inputstring;

}
public void setInputstring(String inputstring) {

this.inputstring = inputstring;

}

public String getLookupstring() {

return lookupstring;

}



public void setLookupstring(String lookupstring) {

this.lookupstring = lookupstring;

}



@Override

public int doStartTag() throws JspException {

try {

JspWriter out = pageContext.getOut(); out.println(StringUtils.countMatches(inputstring,
lookupstring));

} catch (IOException e) { e.printStackTrace();

}

return SKIP_BODY;

}

}
```

In the above code, we have an implementation of the doStartTag() method which is must if we are extending TagSupport class. We have declared two variables inputstring and lookupstring. These variables represents the attributes of the custom tag. We must provide getter and setter for these variables in order to set the values into these variables that will be provided at the time of using this custom tag. We can also specify whether these attributes are required or not.

------------------------------------MyTags.tld---------------------------------------

```
<?xml version="1.0" encoding="UTF-8"?>

<taglib>
```

```
<tlibversion>1.0</tlibversion>

<jspversion>1.1</jspversion>

<shortname>cntmtchs</shortname>

<info>Sample taglib for Substr operation</info>

<uri>http://studytonight.com/jsp/taglib/countmatches</uri>



<tag>

<name>countmatches</name>

<tagclass>com.studytonight.taghandler.CountMatches</tagclass>

<info>String Utility</info>

<attribute>

<name>inputstring</name>

<required>true</required>

</attribute>

<attribute>

<name>lookupstring</name>

<required>true</required>

</attribute>

</tag>

</taglib>
```

The taglib element specifies the schema, required JSP version and the tags within this tag library. Each tag element within the TLD represents an individual custom tag that exist in the library. Each of these tag should have a tag handler class associated with them.

The uri element represents a Uniform Resource Identifier that uniquely identifies the tag library.

The two attribute elements within the tag element represents that the tag has two attributes and the true value provided to the required element represents that both of these attributes are required for the tag to function properly.

--index.jsp------------------------------------

```
<%@taglib prefix="mytag" uri="/WEB-INF/CountMatchesDescriptor.tld"%>

<body>

<h3>Custom Tag Examole</h3>

<mytag:countmatches inputstring="ashoksoft" lookupstring="o" />

</body>



</html>
```

--0---

If this tag works fine it should print a value 3 in the browser as there 't' occurs 2 times in the word "ashoksoft"..

JSP Action Element

Action elements are used to performs specific operation using JSP container, like setting values into java class, getting values from java class. The JSP action elements classified into two types are

- ✓ JSP Standard Action Element
- ✓ JSP Custom Action Element

Standard Action Element

Standard Action are given in JSP to separate the presentation logic and the business logic of the JSP but partial.

The name is given as a standard Action, because each action as a pre-defined meaning and as a programmer it is not possible to change the meaning of the tag.

Each Standard Action follows xml syntax, we do not have any equivalent html syntax.

The standard action element followed by "JSP" prifix.

Syntax of Standard Action Element

<jsp: standard action name>

Standard Action are given by JSP are

- ✓ <jsp: include>
- ✓ <jsp: forward>
- ✓ <jsp: param>
- ✓ <jsp: params>
- ✓ <jsp: plugin>
- ✓ <jsp: useBean>
- ✓ <jsp: setProperty>
- ✓ <jsp: getProperty>
- ✓ <jsp: fallback>

Jsp Forward

This action tag is used for forwarding a request from a jsp to another jsp or a servlet or a html.

When use JSP Forward tag

Generally, if a huge logic is required in a jsp then we divide that logic into multiple jsp pages and then we apply forwarding technique.

Note: If destination is a jsp or html then file name is required and if destination is a servlet then url pattern is requied.

Save and Compile jsp program

JSP program must be save with the .jsp extension. And for compile jsp code simply follow step of compilation servlet code.

For example

forward.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Forward JSP</title>
</head>
<body>
<%-- <jsp:forward page="/welcome"/>
<jsp:forward page="division.html"/>
<jsp:forward page="taglib.jsp"/> --%>
<jsp:forward page="home.jsp">
<jsp:param name="name" value="Praveen"/>
<jsp:param name="age" value="32"/>
</jsp:forward>
</body>
</html>
```

home.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Home Page</title>
</head>
<body>
My name is: <%=request.getParameter("name") %><br>
Age is: <%=request.getParameter("age") %><br>
</body>
</html>
```

Jsp Include

This action tag is used for including the response of one resource like a jsp or a servlet or a html into another jsp page. In jsp, we have two types of including,

- ✓ include directive
- ✓ include action

Main difference between include directive and include action

Include directive is for static including and include action is for dynamic including.

Internally a container uses RequestDispatcher's include method, for executing tag.

	Include Directive	**Include Action**
1	In Include directive, the code of one jsp page is inserted into another jsp. It is called as compile time or static including.	Include Action, the response of one page will be inserted into another page. It is called as run-time or dynamic including.
2	In Include directive, for both source and destination pages, only a single Servlet will be generated internally so number of Servlet object are reduced.	In Include action, individually Servlet are generated for each jsp page. So it increases the number of servlet object.
3	In Include directive, the destination must be jsp page only.	In Include action, the destination resource can be a jsp or Servlet or html.
4	In jsp include directive, we have both html and xml syntax for the tag.	In include action, we have only xml syntax but there is one html syntax.
5	<%@include file=" "%>	<%JSP:include page=" "%>
6	In include directive, the file must be available in the within some application.	In include action the page may exits either within same application or another web application of server.

When use JSP Include Directive

We choose include directive when a destination is a static page like html and we choose include action, when a destination is a dynamic resource like a jsp or a servlet.

For example

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Index Page</title>
</head>
<body>
<jsp:include page="display.jsp">
<jsp:param name="userid" value="Praveen" />
<jsp:param name="password" value="Praveen" />
```

```
<jsp:param name="name" value="Praveen Oruganti" />
<jsp:param name="age" value="32" />
</jsp:include>
</body>
</html>
```

display.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Display Page</title>
</head>
<body>
UserID: <%=request.getParameter("userid") %><br>
Password is: <%=request.getParameter("password") %><br>
User Name: <%=request.getParameter("name") %><br>
Age: <%=request.getParameter("age") %>
</body>
</html>
```

Jsp useBean

<jsp:useBean> standard action tag is use to establish a connection between a jsp page and a java bean.

In web applications of java, jsp and java bean communication is required in the following two cases:

- ✓ In a real-time MVC project, a model class (business class) will set the data to a java bean and a jsp (view) will read the data from a bean and finally displays it on the browser. In this class jsp to a java bean communication is required.
- ✓ If multiple jsp pages need common java logic then it separates that java code into a bean and then we call the bean from jsp. In this case also jsp to java bean communication is required.

<jsp:useBean> tag is used for creating an object of a bean class in a jsp page.

Every java class is not a java bean automatically. A class should have the following qualities to make it as a java bean.

- ✓ Class must be public.
- ✓ Class must contain default constructor.
- ✓ Each private property of the class must have setter or getter or both methods.
- ✓ A class can at most implement Serializable interface.

Syntax

<jsp: useBean id="unique_name_to_identify_bean"

class="package_name.class_name" />

Jsp setProperty

This action tag is to set an input value to set input value to a property on a variable of bean class by calling setter () method of the property.

When use JSP setPriority tag

When a request comes from browser, the protocol is http and it is unknown for the java bean. So, directly input values from browser can't be send to a java bean.

The flow of setting input values is, a jsp page takes request from browser and it will set input values to bean using tag.

<jsp:setProperty> must be used inside <jsp:useBean> tag.

<jsp:setProperty> property contains four attributes, they are;

Name and property attributes are mandatory. We shouldn't use param and value attribute at time.

If request parameters names and variable name of java beans or matched then we can directly write <protperty="*">.

Attribute	Description
Name	Object name: The values of this attribute must be same as the value of id of the bean class
Property	variable name in bean class
Param	request parameter name
Value	static value

Name, property,value	allowed
Name ,property, param	allowed
Property, param	not allowed
Name, property,name, param	not allowed.

Syntax

<jsp:setProperty name="unique_name_to_identify_bean"

property="property_name" />

Jsp getProperty

This action tag is used to read the value of a variable of bean class by calling its getter () method. This action tag must be written at outside of tag.

Attribute of getProperty tag

This action tag has only two attribute called name and property.

In this tag property="*" is not allowed.

Syntax

<jsp:getProperty name="unique_name_to_identify_bean"

property="property_name" />

Example

Details.java

```
package com.praveen.bean;

public class Details {

	public Details() {
	}

	private String username;
	private int age;
	private String password;

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

}
```

usebean.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>UseBean Page</title>
</head>
<body>
<form action="userdetails.jsp" method="post">
User Name: <input type="text" name="username"><br>
User Password: <input type="password" name="password"><br>
User Age: <input type="text" name="age"><br>
```

```
<input type="submit" value="register">
</form>
</body>
</html>
```

userdetails.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>User Details Page</title>
</head>
<body>
<jsp:useBean id="userinfo" class="com.praveen.bean.Details"></jsp:useBean>
<jsp:setProperty property="*" name="userinfo"/>
You have enterted the below details:<br>
<jsp:getProperty property="username" name="userinfo"/><br>
<jsp:getProperty property="password" name="userinfo"/><br>
<jsp:getProperty property="age" name="userinfo" /><br>
</body>
</html>
```

Sample Programs

test1.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
Welcome to JSP
</body>
</html>
```

test2.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
```

```
<b>Welcome to JSP</b> <br>
<i>Welcome to JSP</i> <br>
<u>Welcome to JSP</u> <br>
<u><b>Welcome to JSP </b></u>
</body>
</html>
```

test3.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int i = 100;
i++;
%>
i value is : <%= i %>
</body>
</html>
```

test4.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int i = 100;
i++;
%>
<%
int j = i;
j += 200;
%>
i value is : <%= i %> <br>
j value is : <%= j %>
</body>
</html>
```

test5.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%= 1000 %> <br>
<%= 10.105f %> <br>
<%= 1000.1005 %> <br>
<%= true %> <br>
<%= false %> <br>
<%= 'c' %> <br>
<%= "HELLO" %> <br>
<%= "Hello " + " Buddy" %> <br>
<%= 100+200 %> <br>
<%= 100*200 %>
</body>
</html>
```

test6.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int i =100;
i+= 10;
%>
<%= "i value is : " +i %> <br>
<%
i+= 20;
i++;
%>
<%= "i value is : " +i %> <br>
<%
i+= 30;
i++;
%>
i value is : <%= i %>
</body>
</html>
```

test7.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Insert title here</title>
</head>
<body>
<%
java.util.Date date = new java.util.Date();
String s = "Praveen";
%>
Date is : <%= date %> <br>
s value : <%= s %>
</body>
</html>
```

test8.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int i = 100;
void test()
{
}
%>
i value is : <%= i %>
</body>
</html>
```

test9.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
float f = 200.2f;
float test()
{
return (100+f);
}
%>
f value is : <%= f %> <br>
test() is : <%= test() %>
</body>
</html>
```

test10.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
String s ="Global variable";
%>
S value is : <%=
s
%>
</body>
</html>
```

test11.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
String s ="Global variable";
%>
<%
String s = "Local variable";
%>
S value is : <%=
s
%>
</body>
</html>
```

test12.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
```

```
String s ="Global variable";
%>
<%
String s = "Local variable";
%>
Global S value is : <%= this.s %> <br>
Local S value is : <%= s %>
</body>
</html>
```

test13.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int day = 2;
%>
<%
if(day == 1) { %>
1st Day
<% } else if (day ==2) { %>
2nd Day
<% } else { %>
Some other Day
<% } %>
</body>
</html>
```

test14.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<% int day =3; %>
Switch Statement is used. <br>
<%
switch(day) {
case 1:
out.println("Day 1");
break;
case 2:
out.println("Day 2");
break;
case 3:
out.println("Day 3");
break;
default:
out.println("Some other day");
}
%>
</body>
</html>
```

test15.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
For Loop Example <br> <br>
<%!
int fontSize;
%>
<% for (fontSize = 1; fontSize<3; fontSize++) { %>
<font color = "Red"> <%= fontSize %>
JSP Tutorial <br></font>
<% } %>
</body>
</html>
```

test16.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@ page import = "java.util.Date" %>
<%
Date d = new Date();
%>
<%= "Today Date is : " %>
<%= d %>
</body>
</html>
```

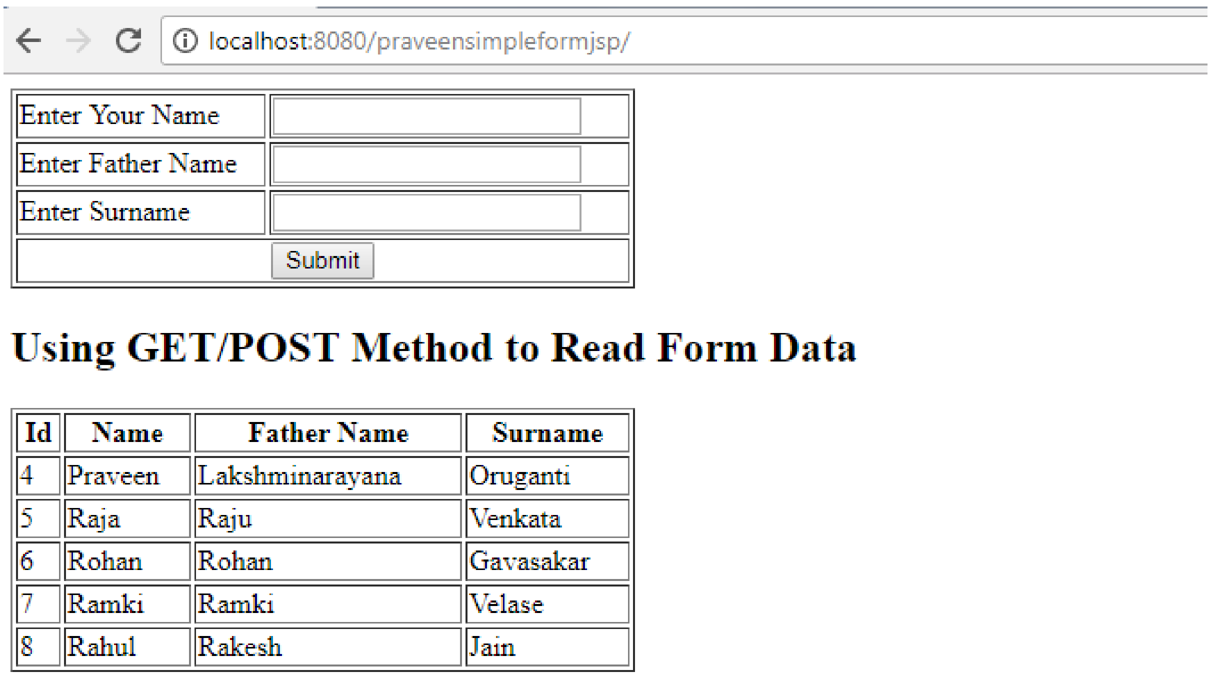
JSP TAGS

1.For Print Data

•<%= "Hello World!" %>

•<% out.println("Welcome in JSP"); %>
2. Declaration
<%!
int radius = 7;
double pi = 3.1415;
%>
3. Script or Logic
•<% pi *= 2; %>
•4. Package
•<%@page import="java.util.Random" %>
• <%= new Random(100)%>

praveensimpleformjsp application



localhost:8080/praveensimpleformjsp/

Enter Your Name	
Enter Father Name	
Enter Surname	
	Submit

Using GET/POST Method to Read Form Data

Id	Name	Father Name	Surname
4	Praveen	Lakshminarayana	Oruganti
5	Raja	Raju	Venkata
6	Rohan	Rohan	Gavasakar
7	Ramki	Ramki	Velase
8	Rahul	Rakesh	Jain

1. query.sql

use Praveen;

```
create table Student (StudentId int NOT NULL
AUTO_INCREMENT,
                      StudentName VARCHAR(20),
                      FatherName VARCHAR(20),
                      surname VARCHAR(20),
                      PRIMARY KEY (StudentId));
select * from Student;
```

2. web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>praveensimpleformjsp</display-name>
  <welcome-file-list>
    <welcome-file>simpleform.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

3. simpleform.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1" import="java.sql.*" errorPage=""%>
<%@page import="java.util.*"%>
<%@include file="database.jsp"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Simple Form Page</title>
</head>
<body>
      <form action="#" method="post">
            <table width="350" border="1">
                  <tr>
                        <td><label>Enter Your Name</label></td>
                        <td><input name="first_name" type="text">&nbsp;</td>
                  </tr>
                  <tr>
                        <td>Enter Father Name</td>
                        <td><input name="last_name" type="text"></td>
                  </tr>
                  <tr>
                        <td>Enter Surname</td>
                        <td><input name="sur_name" type="text"></td>
                  </tr>
                  <tr>
```

```
                        <td colspan="2"><center>
                                <input name="" type="submit">
                            </center></td>
                    </tr>
                </table>
        </form>
        <h2>Using GET/POST Method to Read Form Data</h2>
        <%
                Database db = new Database();
                db.init();
                if (request.getMethod().equalsIgnoreCase("POST")) {
                        try {
                                String Id = request.getParameter("stdId");
                                String name =
request.getParameter("first_name").trim();
                                String fname =
request.getParameter("last_name").trim();
                                String surname =
request.getParameter("sur_name").trim();
                                if (name.length() != 0 && fname.length() != 0) {
                                        db.addStudent(name, fname, surname);
                                }
                                Vector<Bean> vect = db.getStudent();
        %>
        <table width="350" border="1">
                <tr>
                        <th>Id</th>
                        <th>Name</th>
                        <th>Father Name</th>
                        <th>Surname</th>
                </tr>
                <%
                        for (Bean std : vect) {
                %>
                <tr>
                        <td><%=std.Id%></td>
                        <td><%=std.Name%></td>
                        <td><%=std.Fname%></td>
                        <td><%=std.Surname%></td>
                </tr>
                <%
                        }
                %>
        </table>
        <%
                } catch (Exception e) {
                                out.println(e);
                        }
                } //end if
                else if(request.getMethod().equalsIgnoreCase("GET")){

                        Vector<Bean> vect = db.getStudent();
          %>
        <table width="350" border="1">
                <tr>
                        <th>Id</th>
                        <th>Name</th>
                        <th>Father Name</th>
                        <th>Surname</th>
```

```
            </tr>
            <%
             for (Bean std : vect) {
            %>
            <tr>
                   <td><%=std.Id%></td>
                   <td><%=std.Name%></td>
                   <td><%=std.Fname%></td>
                   <td><%=std.Surname%></td>
            </tr>
            <%
              }
             %>
      </table>
      <%
            }

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

4. database.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="java.sql.*" errorPage="" %>
<%@page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Simple Form Page</title>
</head>
<body>
<%!
public class Bean{
public int Id;
public String Name;
public String Fname;
public String Surname;
}
public class Database {
Connection con;
public void init()throws Exception{
String url = "jdbc:mysql://localhost:3306/Praveen";
String user = "root";
String pass = "";
Class.forName("com.mysql.jdbc.Driver");
System.out.println("Driver Loaded Sucessfully");
con = DriverManager.getConnection(url, user, pass);
System.out.println("Connection Created Sucessfully");
}//end init
public Vector<Bean> getStudent()throws SQLException{
Statement stat = null;
ResultSet result = null;
Vector<Bean> vect = new Vector<Bean>();
String query = "Select StudentId, StudentName, FatherName, surname from student";
try{
stat = con.createStatement();
result=stat.executeQuery(query);
while(result.next()){
Bean std = new Bean();
std.Id = result.getInt("StudentId");
std.Name = result.getString("StudentName");
std.Fname = result.getString("FatherName");
std.Surname = result.getString("surname");
vect.addElement(std);
}//end loop
}finally{
if(stat!=null) stat.close();
}
return vect;
}//end getStudent
public int addStudent(String name, String fname, String surname)throws
SQLException{
Statement stat = null;
int rows=0;
String query = "insert into student (StudentName, FatherName, surname)
VALUES('"+name+"','"+fname+"','"+surname+"')";
try{
stat = con.createStatement();
rows = stat.executeUpdate(query);
return rows;
}finally{
if(stat!=null) stat.close();
}
}//end insert
}//end class
%>
</body>
</html>
```