1. **JSP Tutorial**

**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.

A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc.

### **Advantage of JSP over Servlet**

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

#### **1) Extension to Servlet**

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

#### **2) Easy to maintain**

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

#### **3) Fast Development: No need to recompile and redeploy**

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

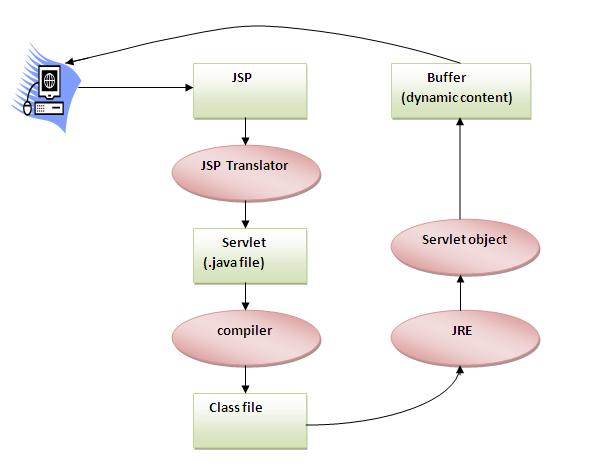
#### **4) Less code than Servlet**

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

### **Life cycle of a JSP Page**

The JSP pages follows these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (class file is loaded by the classloader)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( jspInit() method is invoked by the container).
* Reqeust processing ( \_jspService() method is invoked by the container).
* Destroy ( jspDestroy() method is invoked by the container).



### **The JSP API**

The JSP API consists of two packages:

1. javax.servlet.jsp
2. javax.servlet.jsp.tagext

### **JSP Scriptlet tag (Scripting elements)**

In JSP, java code can be written inside the jsp page using the scriptlet tag.

## **JSP Scripting elements**

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

* scriptlet tag
* expression tag
* declaration tag

### **Example of JSP scriptlet tag that prints the user name**

*File: 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>**

*File: 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 **JSP 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.

*File: index.jsp*

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

*File: welcome.jsp*

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

### **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. **<**%! int data=50; %**>**
4. **<**%= "Value of the variable is:"+data %**>**
5. **</body>**
6. **</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. int cube(int n){
5. return n\*n\*n\*;
6. }
7. %**>**
8. **<**%= "Cube of 3 is:"+cube(3) %**>**
9. **</body>**
10. **</html>**
11. **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.

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 |

# JSP application implicit object

In JSP, application is an implicit object of type *ServletContext*.

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 configuaration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

### **Example of application implicit object:**

**index.html**

1. **<form** action="welcome"**>**
2. **<input** type="text" name="uname"**>**
3. **<input** type="submit" value="go"**><br/>**
4. **</form>**

**web.xml file**

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

**welcome.jsp**

1. **<**%
2. out.print("Welcome "+request.getParameter("uname"));
3. String driver=application.getInitParameter("dname");
4. out.print("driver name is="+driver);
5. %**>**

### **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. String name=request.getParameter("uname");
5. out.print("Welcome "+name);
6. pageContext.setAttribute("user",name,PageContext.SESSION\_SCOPE);
7. <a href="second.jsp">second jsp page</a>
8. %>
9. </body>
10. </html>

### **second.jsp**

1. <html>
2. <body>
3. <%
4. String name=(String)pageContext.getAttribute("user",PageContext.SESSION\_SCOPE);
5. out.print("Hello "+name);
6. %>
7. </body>
8. </html>

### **page implicit object:**

|  |
| --- |
| In JSP, page is an implicit object of type Object class.This object is assigned to the reference of auto generated servlet class. It is written as: |
| 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"); %> |

### **exception implicit object**

|  |
| --- |
| In 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.It is better to learn it after page directive. Let's see a simple example: |

### **Example of exception implicit object:**

### error.jsp

1. <%@ page isErrorPage="true" %>
2. <html>
3. <body>
4. Sorry following exception occured:<%= exception %>
5. </body>
6. </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

### **JSP page directive**

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

### **Syntax of JSP page directive**

1. <%@ 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. <%@ page **import**="java.util.Date" %>
4. Today is: <%= **new** Date() %>
5. </body>
6. </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. <%@ page contentType=application/msword %>
4. Today is: <%= **new** java.util.Date() %>
5. </body>
6. </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. <%@ page info="composed by Sonoo Jaiswal" %>
4. Today is: <%= **new** java.util.Date() %>
5. </body>
6. </html>

### **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. <%@ page buffer="16kb" %>
4. Today is: <%= **new** java.util.Date() %>
5. </body>
6. </html>

### **6)language**

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

### **7)isELIgnored**

|  |
| --- |
| We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. Expression Language is enabled by default. We see Expression Language later. |

1. <%@ page isELIgnored="true" %>//Now EL will be ignored

### **8)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. }

### **9)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. <%@ page errorPage="myerrorpage.jsp" %>
5. <%= 100/0 %>
6. </body>
7. </html>

### **10)isErrorPage**

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

### **Example of isErrorPage attribute**

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

Jsp Include Directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file.

### **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. <%@ include file="header.html" %>
4. Today is: <%= java.util.Calendar.getInstance().getTime() %>
5. </body>
6. </html>

**JSP 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.

### **Example of JSP Taglib directive**

In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

1. <html>
2. <body>
3. <%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %>
4. <mytag:currentDate/>
5. </body>
6. </html>

### **Example of exception handling in jsp by the elements of page directive**

#### **index.jsp**

1. <form action="process.jsp">
2. No1:<input type="text" name="n1" /><br/><br/>
3. No1:<input type="text" name="n2" /><br/><br/>
4. <input type="submit" value="divide"/>
5. </form>

#### **process.jsp**

1. <%@ page errorPage="error.jsp" %>
2. <%
3. String num1=request.getParameter("n1");
4. String num2=request.getParameter("n2");
5. **int** a=Integer.parseInt(num1);
6. **int** b=Integer.parseInt(num2);
7. **int** c=a/b;
8. out.print("division of numbers is: "+c);
9. %>

#### **error.jsp**

1. <%@ page isErrorPage="true" %>
2. <h3>Sorry an exception occured!</h3>
3. Exception is: <%= exception %>
4. **JSP Action Tags**

### **Example of jsp:forward action tag without parameter**

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

### **index.jsp**

1. <html>
2. <body>
3. <h2>**this** is index page</h2>
5. <jsp:forward page="printdate.jsp" />
6. </body>
7. </html>

### **printdate.jsp**

1. <html>
2. <body>
3. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
4. </body>
5. </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**

1. <html>
2. <body>
3. <h2>**this** is index page</h2>
5. <jsp:forward page="printdate.jsp" >
6. <jsp:param name="name" value="javatpoint.com" />
7. </jsp:forward>
9. </body>
10. </html>

### **printdate.jsp**

1. <html>
2. <body>
4. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
5. <%= request.getParameter("name") %>
7. </body>
8. </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.

The jsp:include tag can be used to include static as well as dynamic pages.

### **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*

1. <h2>**this** is index page</h2>
3. <jsp:include page="printdate.jsp" />
5. <h2>end section of index page</h2>

*File: printdate.jsp*

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

**Java Bean**

A Java Bean is a java class that should follow following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

### **Simple example of java bean class**

1. //Employee.java
2. **package** mypack;
3. **public** **class** Employee **implements** java.io.Serializable{
4. **private** **int** id;
5. **private** String name;
6. **public** Employee(){}
7. **public** **void** setId(**int** id){**this**.id=id;}
8. **public** **int** getId(){**return** id;}
9. **public** **void** setName(String name){**this**.name=name;}
10. **public** String getName(){**return** name;}
11. }

### **How to access the java bean class?**

|  |
| --- |
| To access the java bean class, we should use getter and setter methods. |

1. **package** mypack;
2. **public** **class** Test{
3. **public** **static** **void** main(String args[]){
4. Employee e=**new** Employee();//object is created
5. e.setName("Arjun");//setting value to the object
6. System.out.println(e.getName());
7. }}

#### Note: There are two ways to provide values to the object, one way is by constructor and second is by setter method.

# jsp:useBean action tag

The jsp:useBean action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

## **Syntax of jsp:useBean action tag**

1. <jsp:useBean id= "instanceName" scope= "page | request | session | application"
2. **class**= "packageName.className" type= "packageName.className"
3. beanName="packageName.className | <%= expression >" >
4. </jsp:useBean>

### **Simple example of jsp:useBean action tag**

In this example, we are simply invoking the method of the Bean class.

### **Calculator.java (a simple Bean class)**

1. **package** com.javatpoint;
2. **public** **class** Calculator{
3. **public** **int** cube(**int** n){**return** n\*n\*n;}
4. }

### **index.jsp file**

1. <jsp:useBean id="obj" **class**="com.javatpoint.Calculator"/>
2. <%
3. **int** m=obj.cube(5);
4. out.print("cube of 5 is "+m);
5. %>

# jsp:setProperty and jsp:getProperty action tags

The setProperty and getProperty action tags are used for developing web application with Java Bean. In web devlopment, bean class is mostly used because it is a reusable software component that represents data.

The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

## **Syntax of jsp:setProperty action tag**

1. <jsp:setProperty name="instanceOfBean" property= "\*"   |
2. property="propertyName" param="parameterName"  |
3. property="propertyName" value="{ string | <%= expression %>}"
4. />

### **Example of jsp:setProperty action tag if you have to set all the values of incoming request in the bean**

1. <jsp:setProperty name="bean" property="\*" />

### **Example of jsp:setProperty action tag if you have to set value of the incoming specific property**

1. <jsp:setProperty name="bean" property="username" />

### **Example of jsp:setProperty action tag if you have to set a specific value in the property**

1. <jsp:setProperty name="bean" property="username" value="Kumar" />

## **jsp:getProperty action tag**

The jsp:getProperty action tag returns the value of the property.

### **Syntax of jsp:getProperty action tag**

1. <jsp:getProperty name="instanceOfBean" property="propertyName" />

### **Simple example of jsp:getProperty action tag**

1. <jsp:getProperty name="obj" property="name" />

#### **index.html**

1. <form action="process.jsp" method="post">
2. Name:<input type="text" name="name"><br>
3. Password:<input type="password" name="password"><br>
4. Email:<input type="text" name="email"><br>
5. <input type="submit" value="register">
6. </form>

#### **process.jsp**

1. <jsp:useBean id="u" **class**="org.sssit.User"></jsp:useBean>
2. <jsp:setProperty property="\*" name="u"/>
4. Record:<br>
5. <jsp:getProperty property="name" name="u"/><br>
6. <jsp:getProperty property="password" name="u"/><br>
7. <jsp:getProperty property="email" name="u" /><br>

#### **User.java**

1. **package** org.sssit;
3. **public** **class** User {
4. **private** String name,password,email;
5. //setters and getters
6. }

#### **Reusing Bean in Multiple Jsp Pages**

Let's see the simple example, that prints the data of bean object in two jsp pages.

#### **index.jsp-**Same as above

#### **User.java-**Same as above.

#### **process.jsp**

1. <jsp:useBean id="u" **class**="org.sssit.User" scope="session"></jsp:useBean>
2. <jsp:setProperty property="\*" name="u"/>
3. Record:<br>
4. <jsp:getProperty property="name" name="u"/><br>
5. <jsp:getProperty property="password" name="u"/><br>
6. <jsp:getProperty property="email" name="u" /><br>
7. <a href="second.jsp">Visit Page</a>

#### **second.jsp**

1. <jsp:useBean id="u" **class**="org.sssit.User" scope="session"></jsp:useBean>
2. Record:<br>
3. <jsp:getProperty property="name" name="u"/><br>
4. <jsp:getProperty property="password" name="u"/><br>
5. <jsp:getProperty property="email" name="u" /><br>

#### **Using variable value in setProperty tag**

In some case, you may get some value from the database, that is to be set in the bean object, in such case, you need to use expression tag. For example:

#### **process.jsp**

1. <jsp:useBean id="u" **class**="org.sssit.User"></jsp:useBean>
2. <%
3. String name="arjun";
4. %>
5. <jsp:setProperty property="name" name="u" value="<%=name %>"/>
6. Record:<br>
7. <jsp:getProperty property="name" name="u"/><br>

# Expression Language (EL) in JSP

The **Expression Language** (EL) simplifies the accessibility of data stored in the Java Bean component, and other objects like request, session, application etc.

There are many implicit objects, operators and reserve words in EL.

It is the newly added feature in JSP technology version 2.0.

### **Syntax for Expression Language (EL)**

1. ${ expression }

### **Implicit Objects in Expression Language (EL)**

There are many implicit objects in the Expression Language. They are as follows:

|  |  |
| --- | --- |
| **Implicit Objects** | **Usage** |
| pageScope | it maps the given attribute name with the value set in the page scope |
| requestScope | it maps the given attribute name with the value set in the request scope |
| sessionScope | it maps the given attribute name with the value set in the session scope |
| applicationScope | it maps the given attribute name with the value set in the application scope |
| param | it maps the request parameter to the single value |
| paramValues | it maps the request parameter to an array of values |
| header | it maps the request header name to the single value |
| headerValues | it maps the request header name to an array of values |
| cookie | it maps the given cookie name to the cookie value |
| initParam | it maps the initialization parameter |
| pageContext | it provides access to many objects request, session etc. |

### **EL param example**

In this example, we have created two files index.jsp and process.jsp. The index.jsp file gets input from the user and sends the request to the process.jsp which in turn prints the name of the user using EL.

#### **index.jsp**

1. <form action="process.jsp">
2. Enter Name:<input type="text" name="name" /><br/><br/>
3. <input type="submit" value="go"/>
4. </form>

#### **process.jsp**

1. Welcome, ${ param.name }

### **EL sessionScope example**

In this example, we printing the data stored in the session scope using EL. For this purpose, we have used sessionScope object.

#### **index.jsp**

1. <h3>welcome to index page</h3>
2. <%
3. session.setAttribute("user","sonoo");
4. %>
6. <a href="process.jsp">visit</a>

#### **process.jsp**

1. Value is ${ sessionScope.user }

### **EL cookie example**

#### **index.jsp**

1. <h1>First JSP</h1>
2. <%
3. Cookie ck=**new** Cookie("name","abhishek");
4. response.addCookie(ck);
5. %>
6. <a href="process.jsp">click</a>

#### **process.jsp**

1. Hello, ${cookie.name.value}

# JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

## **Advantage of JSTL**

1. **Fast Developement** JSTL provides many tags that simplifies the JSP.
2. **Code Reusability** We can use the JSTL tags in various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag

#### For creating JSTL application, you need to load jstl.jar file.

## **JSTL Core Tags List**

# <c:out> Tag

The <c:out> tag is similar to JSP expression tag, but it can only be used with expression. It will display the result of an expression, similar to the way < %=...% > work.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:out** value="${'Welcome to javaTpoint'}"**/>**
8. **</body>**
9. **</html>**

# <c:import> Tag

The <c:import> is similar to jsp 'include', with an additional feature of including the content of any resource either within server or outside the server.

This tag provides all the functionality of the <include > action and it also allows the inclusion of absolute URLs.

For example: Using an import tag the content from a different FTP server and website can be accessed.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:import** var="data" url="http://www.javatpoint.com"**/>**
8. **<c:out** value="${data}"**/>**
9. **</body>**
10. **</html>**

# <c:set> Tag

It is used to set the result of an expression evaluated in a 'scope'. The <c:set> tag is helpful because it evaluates the expression and use the result to set a value of java.util.Map or JavaBean.

This tag is similar to jsp:setProperty action tag.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:set** var="Income" scope="session" value="${4000\*4}"**/>**
8. **<c:out** value="${Income}"**/>**
9. **</body>**
10. **</html>**

# <c:remove> Tag

It is used for removing the specified variable from a particular scope. This action is not particularly helpful, but it can be used for ensuring that a JSP can also clean up any scope resources.

The <c:remove > tag removes the variable from either a first scope or a specified scope.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:set** var="income" scope="session" value="${4000\*4}"**/>**
8. **<p>**Before Remove Value is: **<c:out** value="${income}"**/></p>**
9. **<c:remove** var="income"**/>**
10. **<p>**After Remove Value is: **<c:out** value="${income}"**/></p>**
11. **</body>**
12. **</html>**

Output:

1. Before Remove Value is: 16000
2. After Remove Value is:

# <c:catch> Tag

It is used for Catches any Throwable exceptions that occurs in the body and optionally exposes it. In general it is used for error handling and to deal more easily with the problem occur in program.

The < c:catch > tag catches any exceptions that occurs in a program body.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
8. **<c:catch** var ="catchtheException"**>**
9. **<**% int x = 2/0;%**>**
10. **</c:catch>**
12. **<c:if** test = "${catchtheException != null}"**>**
13. **<p>**The type of exception is : ${catchtheException} **<br** **/>**
14. There is an exception: ${catchtheException.message}**</p>**
15. **</c:if>**
17. **</body>**
18. **</html>**

Output:

1. The type of exception is : java.lang.ArithmaticException: / by zero
2. There is an exception: / by zero

# <c:if> Tag

The < c:if > tag is used for testing the condition and it display the body content, if the expression evaluated is true.

It is a simple conditional tag which is used for evaluating the body content, if the supplied condition is true.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:set** var="income" scope="session" value="${4000\*4}"**/>**
8. **<c:if** test="${income > 8000}"**>**
9. **<p>**My income is: **<c:out** value="${income}"**/><p>**
10. **</c:if>**
11. **</body>**
12. **</html>**

Output:

1. My income is: 16000

# <c:choose>, <c:when>, <c:otherwise> Tag

The < c:choose > tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch**statement in which we choose between a numbers of alternatives.

The <c:when > is subtag of <choose > that will include its body if the condition evaluated be 'true'.

The < c:otherwise > is also subtag of < choose > it follows &l;twhen > tags and runs only if all the prior condition evaluated is 'false'.

The c:when and c:otherwise works like if-else statement. But it must be placed inside c:choose tag.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:set** var="income" scope="session" value="${4000\*4}"**/>**
8. **<p>**Your income is : **<c:out** value="${income}"**/></p>**
9. **<c:choose>**
10. **<c:when** test="${income <= 1000}"**>**
11. Income is not good.
12. **</c:when>**
13. **<c:when** test="${income > 10000}"**>**
14. Income is very good.
15. **</c:when>**
16. **<c:otherwise>**
17. Income is undetermined...
18. **</c:otherwise>**
19. **</c:choose>**
20. **</body>**
21. **</html>**

This will produce the following result:

1. Your income is : 16000
2. Income is very good.

# <c:forEach> Tag

The <c:for each > is an iteration tag used for repeating the nested body content for fixed number of times or over the collection.

These tag used as a good alternative for embedding a Java **while, do-while, or for**loop via a scriptlet. The < c:for each > tag is most commonly used tag because it iterates over a collection of object.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:forEach** var="j" begin="1" end="3"**>**
8. Item **<c:out** value="${j}"**/><p>**
9. **</c:forEach>**
10. **</body>**
11. **</html>**

Output:

1. Item 1
2. Item 2
3. Item 3

# <c:forTokens> Tag

The < c:forTokens > tag iterates over tokens which is separated by the supplied delimeters. It is used for break a string into tokens and iterate through each of the tokens to generate output.

This tag has similar attributes as < c:forEach > tag except one additional attributes **delims**which is used for specifying the characters to be used as delimiters.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:forTokens** items="Rahul-Nakul-Rajesh" delims="-" var="name"**>**
8. **<c:out** value="${name}"**/><p>**
9. **</c:forTokens>**
10. **</body>**
11. **</html>**

Output:

1. Rahul
2. Nakul
3. Rajesh

# <c:param> Tag

The < c:param > tag add the parameter in a containing 'import' tag's URL. It allow the proper URL request parameter to be specified within URL and it automatically perform any necessary URL encoding.

Inside < c:param > tag, the value attribute indicates the parameter value and name attribute indicates the parameter name.

Let's see the simple example of tag:

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:url** value="/index1.jsp" var="completeURL"**/>**
8. **<c:param** name="trackingId" value="786"**/>**
9. **<c:param** name="user" value="Nakul"**/>**
10. **</c:url>**
11. ${completeURL}
12. **</body>**
13. **</html>**

Output:

1. /JSP/index1.jsp?trackingId=786&user=Nakul

# <c:redirect> Tag

The < c:redirect > tag redirects the browser to a new URL. It supports the context-relative URLs, and the < c:param > tag.

It is used for redirecting the browser to an alternate URL by using automatic URL rewriting.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:set** var="url" value="0" scope="request"**/>**
8. **<c:if** test="${url<1}"**>**
9. **<c:redirect** url="http://javatpoint.com"**/>**
10. **</c:if>**
11. **<c:if** test="${url>1}"**>**
12. **<c:redirect** url="http://facebook.com"**/>**
13. **</c:if>**
14. **</body>**
15. **</html>**

Output:

Since the value of the variable **'url'** is 0 the page gets directed to the **http://javatpoint.com**.

# <c:url> Tag

The < c:url > tag creates a URL with optional query parameter. It is used for url encoding or url formatting. This tag automatically performs the URL rewriting operation.

The JSTL url tag is used as an alternative method of writing call to the response.encodeURL() method. The advantage of url tag is proper URL encoding and including the parameters specified by children. **param**tag.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<html>**
3. **<head>**
4. **<title>**Core Tag Example**</title>**
5. **</head>**
6. **<body>**
7. **<c:url** value="/RegisterDao.jsp"**/>**
8. **</body>**
9. **</html>**

Output:

1. /CRUD/RegisterDao.jsp

# JSTL Function Tags

The JSTL function provides a number of standard functions, most of these functions are common string manipulation functions. The syntax used for including JSTL function library in your JSP is:

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**

# fn:contains() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="String" value="Welcome to javatpoint"**/>**
9. **<c:if** test="${fn:contains(String, 'javatpoint')}"**>**
10. **<p>**Found javatpoint string**<p>**
11. **</c:if>**
12. **<c:if** test="${fn:contains(String, 'JAVATPOINT')}"**>**
13. **<p>**Found JAVATPOINT string**<p>**
14. **</c:if>**
15. **</body>**
16. **</html>**

Output:

1. Found javatpoint string

# fn:containsIgnoreCase() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="String" value="Welcome to javatpoint"**/>**
9. **<c:if** test="${fn:containsIgnoreCase(String, 'javatpoint')}"**>**
10. **<p>**Found javatpoint string**<p>**
11. **</c:if>**
12. **<c:if** test="${fn:containsIgnoreCase(String, 'JAVATPOINT')}"**>**
13. **<p>**Found JAVATPOINT string**<p>**
14. **</c:if>**
15. **</body>**
16. **</html>**

Output:

1. Found javatpoint string
2. Found JAVATPOINT string

# fn:endsWith() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="String" value="Welcome to JSP programming"**/>**
9. **<c:if** test="${fn:endsWith(String, 'programming')}"**>**
10. **<p>**String ends with programming**<p>**
11. **</c:if>**
12. **<c:if** test="${fn:endsWith(String, 'JSP')}"**>**
13. **<p>**String ends with JSP**<p>**
14. **</c:if>**
15. **</body>**
16. **</html>**

Output:

1. String ends with programming

# fn:escapeXml() Function

The fn:escapeXml() function escapes the characters that would be interpreted as XML markup. It is used for escaping the character in XML markup language.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string1" value="It is first String."**/>**
9. **<c:set** var="string2" value="It is <xyz>second String.</xyz>"**/>**
10. **<p>**With escapeXml() Function:**</p>**
11. **<p>**string-1 : ${fn:escapeXml(string1)}**</p>**
12. **<p>**string-2 : ${fn:escapeXml(string2)}**</p>**
13. **<p>**Without escapeXml() Function:**</p>**
14. **<p>**string-1 : ${string1}**</p>**
15. **<p>**string-2 : ${string2}**</p>**
16. **</body>**
17. **</html>**

Output:

1. With escapeXml() Function:
2. string-1 : It is first String.
3. string-2 : It is **<xyz>**second String.**</xyz>**
4. Without escapeXml() Function:
5. string-1 : It is first String.
6. string-2 : It is second String.

# fn:indexOf() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string1" value="It is first String."**/>**
9. **<c:set** var="string2" value="It is <xyz>second String.</xyz>"**/>**
10. **<p>**Index-1 : ${fn:indexOf(string1, "first")}**</p>**
11. **<p>**Index-2 : ${fn:indexOf(string2, "second")}**</p>**
12. **</body>**
13. **</html>**

Output:

1. Index-1 : 6
2. Index-2 : 11

# fn:trim() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="str1" value="Welcome to JSP        programming         "**/>**
9. **<p>**String-1 Length is : ${fn:length(str1)}**</p>**
10. **<c:set** var="str2" value="${fn:trim(str1)}" **/>**
11. **<p>**String-2 Length is : ${fn:length(str2)}**</p>**
12. **<p>**Final value of string is : ${str2}**</p>**
13. **</body>**
14. **</html>**

Output:

1. String-1 Length is : 42
2. String-2 Length is : 33
3. Final value of string is : Welcome to JSP programming

# fn:startsWith() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="msg" value="The Example of JSTL fn:startsWith() Function"**/>**
9. The string starts with "The": ${fn:startsWith(msg, 'The')}
10. **<br>**The string starts with "Example": ${fn:startsWith(msg, 'Example')}
11. **</body>**
12. **</html>**

Output:

1. The string starts with "The": true
2. The string starts with "Example": false

# fn:split() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Functions**</title>**
6. **</head>**
7. **<body>**
9. **<c:set** var="str1" value="Welcome-to-JSP-Programming."**/>**
10. **<c:set** var="str2" value="${fn:split(str1, '-')}" **/>**
11. **<c:set** var="str3" value="${fn:join(str2, ' ')}" **/>**
12. **<p>**String-3 : ${str3}**</p>**
13. **<c:set** var="str4" value="${fn:split(str3, ' ')}" **/>**
14. **<c:set** var="str5" value="${fn:join(str4, '-')}" **/>**
15. **<p>**String-5 : ${str5}**</p>**
16. **</body>**
17. **</html>**

Output:

1. String-3 : Welcome to JSP Programming.
2. String-5 : Welcome-to-JSP-Programming.

# fn:toLowerCase() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>** Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string" value="Welcome to JSP Programming"**/>**
9. ${fn:toLowerCase("HELLO,")}
10. ${fn:toLowerCase(string)}
11. **</body>**
12. **</html>**

Output:

1. hello, welcome to jsp programming

# fn:toUpperCase() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="site" value="javatpoint.com"**/>**
9. **<c:set** var="author" value="Sonoo Jaiswal"**/>**
10. Hi, This is ${fn:toUpperCase(site)} developed by ${fn:toUpperCase(author)}.
11. **</body>**
12. **</html>**

Output:

1. Hi, This is JAVATPOINT.COM developed by SONOO JAISWAL.

# fn:substring() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string" value="This is the first string."**/>**
9. ${fn:substring(string, 5, 17)}
10. **</body>**
11. **</html>**

Output:

1. is the first

# fn:substringAfter() Function

The fn:substringAfter() function returns the subset of string followed by a specific substring. It returns the part of string which lies after the provided string value.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string" value="Nakul Jain"**/>**
9. ${fn:substringAfter(string, "Nakul")}
10. **</body>**
11. **</html>**

Output:

1. Jain

# fn:substringBefore() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="string" value="Hi, This is JAVATPOINT.COM developed by SONOO JAISWAL."**/>**
9. ${fn:substringBefore(string, "developed")}
10. **</body>**
11. **</html>**

Output:

1. Hi, This is JAVATPOINT.COM

# fn:length() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**JSTL fn:length() example**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="str1" value="This is first string"**/>**
9. **<c:set** var="str2" value="Hello"**/>**
10. Length of the String-1 is: ${fn:length(str1)}**<br>**
11. Length of the String-2 is: ${fn:length(str2)}
12. **</body>**
13. **</html>**

Output:

1. Length of the String-1 is: 20
2. Length of the String-2 is: 5

# fn:replace() Function

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %**>**
3. **<html>**
4. **<head>**
5. **<title>**Using JSTL Function **</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="author" value="Ramesh Kumar"**/>**
9. **<c:set** var="string" value="pqr xyz abc PQR"**/>**
10. ${fn:replace(author, "Ramesh", "Suresh")}
11. ${fn:replace(string, "pqr", "hello")}
12. **</body>**
13. **</html>**

Output:

1. Suresh Kumar hello xyz abc PQR

# JSTL Formatting tags

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is**http://java.sun.com/jsp/jstl/fmt**and prefix is **fmt.**

# <fmt:parseNumber> Tag

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %**>**
4. **<html>**
5. **<head>**
6. **<title>**fmt:parseNumber tag**</title>**
7. **</head>**
8. **<body>**
9. **<h3>**The fmt:parseNumber tag Example is:**</h3>**
10. **<c:set** var="Amount" value="786.970" **/>**
11. **<fmt:parseNumber** var="j" type="number" value="${Amount}" **/>**
12. **<p><i>**Amount is:**</i>**  **<c:out** value="${j}" **/></p>**
13. **<fmt:parseNumber** var="j" integerOnly="true" type="number" value="${Amount}" **/>**
14. **<p><i>**Amount is:**</i>**  **<c:out** value="${j}" **/></p>**
15. **</body>**
16. **</html>**

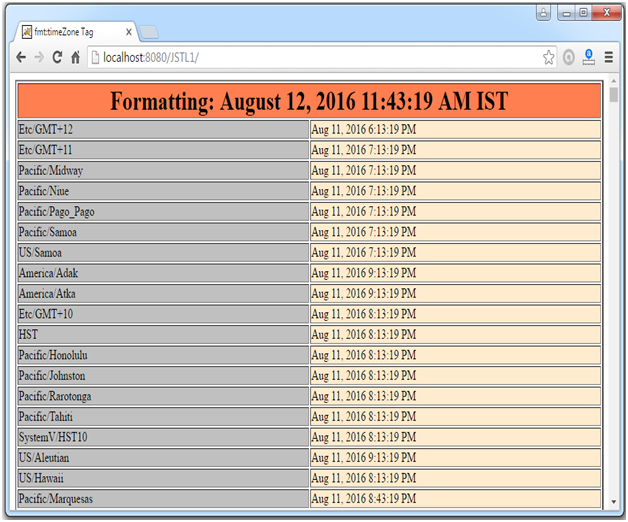
Output:

1. The fmt:parseNumber tag Example is:
2. Amount is: 786.97
3. Amount is: 786

# <fmt:timeZone> Tag

The <fmt:timeZone> tag specifies the parsing action nested in its body or the time zone for any time formatting. It is used for specify the time zone information used for time formatting operations.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %**>**
3. **<html>**
4. **<head>**
5. **<title>**fmt:timeZone Tag**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="str" value="<%=new java.util.Date()%>" **/>**
9. **<table** border="2" width="100%"**>**
10. **<tr>**
11. **<td** width="100%" colspan="2" bgcolor="#FF7F50"**>**
12. **<p** align="center"**>**
13. **<b>**
14. **<font** color="#000000" size="6"**>**Formatting:
15. **<fmt:formatDate** value="${str}" type="both"
16. timeStyle="long" dateStyle="long" **/>**
17. **</font>**
18. **</b>**
19. **</p>**
20. **</td>**
21. **</tr>**
23. **<c:forEach** var="zone"
24. items="<%=java.util.TimeZone.getAvailableIDs()%>"**>**
25. **<tr>**
26. **<td** width="50%"  bgcolor="#C0C0C0"**>**
27. **<c:out** value="${zone}" **/>**
28. **</td>**
29. **<td** width="50%" bgcolor="#FFEBCD"**>**
30. **<fmt:timeZone** value="${zone}"**>**
31. **<fmt:formatDate** value="${str}" timeZone="${zn}"
32. type="both"**/>**
33. **</fmt:timeZone>**
34. **</td>**
35. **</tr>**
36. **</c:forEach>**
37. **</table>**
38. **</body>**
39. **</html>**



# <fmt:formatNumber> Tag

The <fmt:formatNumber> tag is used to format the numerical value using the specific format or precision. It is used to format percentages, currencies, and numbers according to the customized formatting pattern.

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%**>**
2. **<**%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%**>**
4. **<html>**
5. **<head>**
6. **<title>**fmt:formatNumber Tag**</title>**
7. **</head>**
8. **<body>**
9. **<h3>**Formatting of Number:**</h3>**
10. **<c:set** var="Amount" value="9850.14115" **/>**
11. **<p>** Formatted Number-1:
12. **<fmt:formatNumber** value="${Amount}" type="currency" **/></p>**
13. **<p>**Formatted Number-2:
14. **<fmt:formatNumber** type="number" groupingUsed="true" value="${Amount}" **/></p>**
15. **<p>**Formatted Number-3:
16. **<fmt:formatNumber** type="number" maxIntegerDigits="3" value="${Amount}" **/></p>**
17. **<p>**Formatted Number-4:
18. **<fmt:formatNumber** type="number" maxFractionDigits="6" value="${Amount}" **/></p>**
19. **<p>**Formatted Number-5:
20. **<fmt:formatNumber** type="percent" maxIntegerDigits="4" value="${Amount}" **/></p>**
21. **<p>**Formatted Number-6:
22. **<fmt:formatNumber** type="number" pattern="###.###$" value="${Amount}" **/></p>**
23. **</body>**
24. **</html>**

Output:

1. Formatting of Number:
2. Formatted Number-1: $9,850.14
3. Formatted Number-2: 9,850.141
4. Formatted Number-3: 850.141
5. Formatted Number-4: 9,850.14115
6. Formatted Number-5: 5,014%
7. Formatted Number-6: 9850.141$

# Formatting <fmt:parseDate> Tag

The <fmt:parseDate> tag parses the string representation of a time and date. It is used to format the time and date according to a customized formatting pattern.

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %**>**
3. **<html>**
4. **<head>**
5. **<title>**fmt:parseDate Tag**</title>**
6. **</head>**
7. **<body>**
8. **<h3>**Parsed Date:**</h3>**
9. **<c:set** var="date" value="12-08-2016" **/>**
10. **<fmt:parseDate** value="${date}" var="parsedDate"  pattern="dd-MM-yyyy" **/>**
11. **<p><c:out** value="${parsedDate}" **/></p>**
12. **</body>**
13. **</html>**

Output:

**Parsed Date:**

Fri Aug 12 00:00:00 IST 2016

# <fmt:bundle> Tag

The <fmt:bundle> tag loads the resource bundle which is used by its tag body. This tag will make the specified bundle available for all <fmt:message> tags that occurs between the boundary of <fmt:bundle> and </fmt:bundle> tags.

It is used to create the ResourceBundle objects which will be used by their tag body.

**Simple.java**

1. **package** com.javatpoint;
2. **import** java.util.ListResourceBundle;
3. **public** **class** Simple **extends** ListResourceBundle {
4. **public** Object[][] getContents() {
5. **return** contents;
6. }
7. **static** **final** Object[][] contents = { { "colour.Violet", "Violet" },
8. { "colour.Indigo", "Indigo" }, { "colour.Blue", "Blue" }, };
9. }

Now, compile the above class as **Simple.class**and make it available in **CLASSPATH**of your **Web application**folder.

Now you can use the below JSTL tags to display the three colors as follows:

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %**>**
2. **<html>**
3. **<head>**
4. **<title>**fmt:bundle Tag**</title>**
5. **</head>**
6. **<body>**
7. **<fmt:bundle** basename="com.javatpoint.Simple" prefix="colour."**>**
8. **<fmt:message** key="Violet"**/><br/>**
9. **<fmt:message** key="Indigo"**/><br/>**
10. **<fmt:message** key="Blue"**/><br/>**
11. **</fmt:bundle>**
12. **</body>**
13. **</html>**

Output:

1. Violet
2. Indigo
3. Blue

# <fmt:setTimeZone> Tag

The <fmt:setTimeZone> tag store the time zone inside a time zone configuration variable. It is used for copy a time zone object inside a specified scope variable.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %**>**
2. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %**>**
3. **<html>**
4. **<head>**
5. **<title>**fmt:setTimeZone Tag**</title>**
6. **</head>**
7. **<body>**
8. **<c:set** var="date" value="<%=new java.util.Date()%>" **/>**
9. **<p><b>**Date and Time in Indian Standard Time(IST) Zone:**</b>** **<fmt:formatDate** value="${date}"
10. type="both" timeStyle="long" dateStyle="long" **/></p>**
11. **<fmt:setTimeZone** value="GMT-10" **/>**
12. **<p><b>**Date and Time in GMT-10 time Zone: **</b><fmt:formatDate** value="${date}"
13. type="both" timeStyle="long" dateStyle="long" **/></p>**
14. **</body>**
15. **</html>**

Output:

1. Date and Time in Indian Standard Time(IST) Zone:**</strong>** August 12, 2016 6:27:04 PM IST
2. Date and Time in GMT-10 time Zone:**</strong>** August 12, 2016 2:57:04 AM GMT-10:00

# <fmt:setBundle> Tag

The <fmt:setBundle> tag is used to load the resource bundle and store their value in the bundle configuration variable or the name scope variable.

It is used for creating the ResourceBundle object which will be used by tag body.

**Main.java**

1. **package** com.javatpoint;
2. **import** java.util.ListResourceBundle;
3. **public** **class** Main **extends** ListResourceBundle {
4. **public** Object[][] getContents() {
5. **return** contents;
6. }
7. **static** **final** Object[][] contents = { { "vegetable.Potato", "Potato" },
8. { "vegetable.Tomato", "Tomato" }, { "vegetable.Carrot", "Carrot" }, };
9. }

Now, compile the above class as **Main.class**and make it available in **CLASSPATH**of your **Web application**folder.

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %**>**
2. **<html>**
3. **<head>**
4. **<title>**fmt:setBundle Tag**</title>**
5. **</head>**
6. **<body>**
7. **<fmt:setBundle** basename="com.javatpoint.Main" var="lang"**/>**
8. **<fmt:message** key="vegetable.Potato" bundle="${lang}"**/><br/>**
9. **<fmt:message** key="vegetable.Tomato" bundle="${lang}"**/><br/>**
10. **<fmt:message** key="vegetable.Carrot" bundle="${lang}"**/><br/>**
11. **</body>**
12. **</html>**

Output:

1. Potato
2. Tomato
3. Carrot

# <fmt:message> Tag

**Message.java**

1. **package** com.javatpoint;
2. **import** java.util.ListResourceBundle;
3. **public** **class** Message **extends** ListResourceBundle {
4. **public** Object[][] getContents() {
5. **return** contents;
6. }
7. **static** **final** Object[][] contents = { { "vegetable.Potato", "Potato" },
8. { "vegetable.Tomato", "Tomato" }, { "vegetable.Carrot", "Carrot" }, };
9. }

Now, compile the above class as **Message.class**and make it available in **CLASSPATH**of your **Web application**folder.

Now you can use the below JSTL tags to display the three vegetables as follows:

1. **<**%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %**>**
2. **<html>**
3. **<head>**
4. **<title>**fmt:message Tag**</title>**
5. **</head>**
6. **<body>**
7. **<fmt:setBundle** basename="com.javatpoint.Message" var="lang"**/>**
8. **<fmt:message** key="vegetable.Potato" bundle="${lang}"**/><br/>**
9. **<fmt:message** key="vegetable.Tomato" bundle="${lang}"**/><br/>**
10. **<fmt:message** key="vegetable.Carrot" bundle="${lang}"**/><br/>**
11. **</body>**
12. **</html>**

Output:

1. Potato
2. Tomato
3. Carrot

# <fmt:formatDate> Tag

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%**>**
2. **<**%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%**>**
3. **<html>**
4. **<head>**
5. **<title>**fmt:formatDate**</title>**
6. **</head>**
7. **<body>**
8. **<h2>**Different Formats of the Date**</h2>**
9. **<c:set** var="Date" value="<%=new java.util.Date()%>" **/>**
10. **<p>**
11. Formatted Time :
12. **<fmt:formatDate** type="time" value="${Date}" **/>**
13. **</p>**
14. **<p>**
15. Formatted Date :
16. **<fmt:formatDate** type="date" value="${Date}" **/>**
17. **</p>**
18. **<p>**
19. Formatted Date and Time :
20. **<fmt:formatDate** type="both" value="${Date}" **/>**
21. **</p>**
22. **<p>**
23. Formatted Date and Time in short style :
24. **<fmt:formatDate** type="both" dateStyle="short" timeStyle="short"
25. value="${Date}" **/>**
26. **</p>**
27. **<p>**
28. Formatted Date and Time in medium style :
29. **<fmt:formatDate** type="both" dateStyle="medium" timeStyle="medium"
30. value="${Date}" **/>**
31. **</p>**
32. **<p>**
33. Formatted Date and Time in long style :
34. **<fmt:formatDate** type="both" dateStyle="long" timeStyle="long"
35. value="${Date}" **/>**
36. **</p>**
38. **</body>**
39. **</html>**

Output:

1. Different Formats of the Date
2. Formatted Time : 4:20:50 PM
3. Formatted Date : Aug 13, 2016
4. Formatted Date and Time : Aug 13, 2016 4:20:50 PM
5. Formatted Date and Time in short style : 8/13/16 4:20 PM
6. Formatted Date and Time in medium style : Aug 13, 2016 4:20:50 PM
7. Formatted Date and Time in long style : August 13, 2016 4:20:50 PM IST

# JSTL XML tags

# <x:out> Tag

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
3. **<html>**
4. **<head>**
5. **<title>**XML Tags**</title>**
6. **</head>**
7. **<body>**
8. **<h2>**Vegetable Information:**</h2>**
9. **<c:set** var="vegetable"**>**
10. **<vegetables>**
11. **<vegetable>**
12. **<name>**onion**</name>**
13. **<price>**40/kg**</price>**
14. **</vegetable>**
15. **<vegetable>**
16. **<name>**Potato**</name>**
17. **<price>**30/kg**</price>**
18. **</vegetable>**
19. **<vegetable>**
20. **<name>**Tomato**</name>**
21. **<price>**90/kg**</price>**
22. **</vegetable>**
23. **</vegetables>**
24. **</c:set>**
25. **<x:parse** xml="${vegetable}" var="output"**/>**
26. **<b>**Name of the vegetable is**</b>**:
27. **<x:out** select="$output/vegetables/vegetable[1]/name" **/><br>**
28. **<b>**Price of the Potato is**</b>**:
29. **<x:out** select="$output/vegetables/vegetable[2]/price" **/>**
30. **</body>**
31. **</html>**

Output:

## Vegetable Information:

**Name of the vegetable is**: onion

**Price of the Potato is**: 30/kg

# <x:parse> Tag

The <x:parse> tag is used for parse the XML data specified either in the tag body or an attribute. It is used for parse the xml content and the result will stored inside specified variable.

**novels.xml**

1. **<books>**
2. **<book>**
3. **<name>**Three mistakes of my life**</name>**
4. **<author>**Chetan Bhagat**</author>**
5. **<price>**200**</price>**
6. **</book>**
7. **<book>**
8. **<name>**Tomorrow land**</name>**
9. **<author>**NUHA**</author>**
10. **<price>**2000**</price>**
11. **</book>**
12. **</books>**

 index.jsp

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
3. **<html>**
4. **<head>**
5. **<title>**x:parse Tag**</title>**
6. **</head>**
7. **<body>**
8. **<h2>**Books Info:**</h2>**
9. **<c:import** var="bookInfo" url="novels.xml"**/>**
11. **<x:parse** xml="${bookInfo}" var="output"**/>**
12. **<p>**First Book title: **<x:out** select="$output/books/book[1]/name" **/></p>**
13. **<p>**First Book price: **<x:out** select="$output/books/book[1]/price" **/></p>**
14. **<p>**Second Book title: **<x:out** select="$output/books/book[2]/name" **/></p>**
15. **<p>**Second Book price: **<x:out** select="$output/books/book[2]/price" **/></p>**
16. **</body>**
17. **</html>**

Output:

## Books Info:

First Book title: Three mistakes of my life

First Book price: 200

Second Book title: Tomorrow land

Second Book price: 2000

# <x:set> Tag

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
3. **<html>**
4. **<head>**
5. **<title>**x:set Tag**</title>**
6. **</head>**
7. **<body>**
8. **<h3>**Books Information:**</h3>**
9. **<c:set** var="book"**>**
10. **<books>**
11. **<book>**
12. **<name>**Three mistakes of my life**</name>**
13. **<author>**Chetan Bhagat**</author>**
14. **<price>**200**</price>**
15. **</book>**
16. **<book>**
17. **<name>**Tomorrow land**</name>**
18. **<author>**Brad Bird**</author>**
19. **<price>**2000**</price>**
20. **</book>**
21. **</books>**
22. **</c:set>**
23. **<x:parse** xml="${book}" var="output"**/>**
24. **<x:set** var="fragment" select="$output/books/book[2]/price"**/>**
25. **<b>**The price of the Tomorrow land book**</b>**:
26. **<x:out** select="$fragment" **/>**
27. **</body>**
28. **</html>**

Output:

### Books Information:

**The price of the Tomorrow land book**: 2000

# <x:choose>, <x:when>, <x:otherwise> Tags

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
4. **<html>**
5. **<head>**
6. **<title>**x:choose Tag**</title>**
7. **</head>**
8. **<body>**
9. **<h3>**Books Information:**</h3>**
11. **<c:set** var="xmltext"**>**
12. **<books>**
13. **<book>**
14. **<name>**Three mistakes of my life**</name>**
15. **<author>**Chetan Bhagat**</author>**
16. **<price>**200**</price>**
17. **</book>**
18. **<book>**
19. **<name>**Tomorrow land**</name>**
20. **<author>**Brad Bird**</author>**
21. **<price>**2000**</price>**
22. **</book>**
23. **</books>**
24. **</c:set>**
26. **<x:parse** xml="${xmltext}" var="output"**/>**
27. **<x:choose>**
28. **<x:when** select="$output//book/author = 'Chetan bhagat'"**>**
29. Book is written by Chetan bhagat
30. **</x:when>**
31. **<x:when** select="$output//book/author = 'Brad Bird'"**>**
32. Book is written by Brad Bird
33. **</x:when>**
34. **<x:otherwise>**
35. The author is unknown...
36. **</x:otherwise>**
37. **</x:choose>**
39. **</body>**
40. **</html>**

Output:

### Books Information:

Book is written by Brad Bird

# <x:if> Tag

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
3. **<html>**
4. **<head>**
5. **<title>**x:if Tags**</title>**
6. **</head>**
7. **<body>**
8. **<h2>**Vegetable Information:**</h2>**
9. **<c:set** var="vegetables"**>**
10. **<vegetables>**
11. **<vegetable>**
12. **<name>**onion**</name>**
13. **<price>**40**</price>**
14. **</vegetable>**
15. **<vegetable>**
16. **<name>**Potato**</name>**
17. **<price>**30**</price>**
18. **</vegetable>**
19. **<vegetable>**
20. **<name>**Tomato**</name>**
21. **<price>**90**</price>**
22. **</vegetable>**
23. **</vegetables>**
24. **</c:set>**
25. **<x:parse** xml="${vegetables}" var="output"**/>**
26. **<x:if** select="$output/vegetables/vegetable/price < 100"**>**
27. Vegetables prices are very low.
28. **</x:if>**
29. **</body>**
30. **</html>**

Output:

## Vegetable Information:

Vegetables prices are very low.

# <x:transform> Tag

The <x:transform> tag is used in a XML document for providing the XSL (Extensible Stylesheet Language) transformation. It is used for transforming xml data based on XSLT script.

Let us put the following program in **transfer.xsl** file:

1. **<xsl:stylesheet** version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"**>**
2. **<xsl:param** name="doc"**/>**
3. **<xsl:template** match="/"**>**
4. **<html>**
5. **<body>**
6. **<h2>**Company's Employee detail**</h2>**
7. **<table** border="2"**>**
8. **<tr>**
9. **<th** align="left"**>**Name
10. **</th>**
11. **<th** align="left"**>**Designation
12. **</th>**
13. **<th** align="left"**>**Age
14. **</th>**
15. **</tr>**
16. **<xsl:for-each** select="organisation/company/emp"**>**
17. **<tr>**
18. **<td>**
19. **<xsl:value-of** select="name"**/>**
20. **</td>**
21. **<td>**
22. **<xsl:value-of** select="designation"**/>**
23. **</td>**
24. **<td>**
25. **<xsl:value-of** select="age"**/>**
26. **</td>**
27. **</tr>**
28. **</xsl:for-each>**
29. **</table>**
30. **</body>**
31. **</html>**
32. **</xsl:template>**
33. **</xsl:stylesheet>**

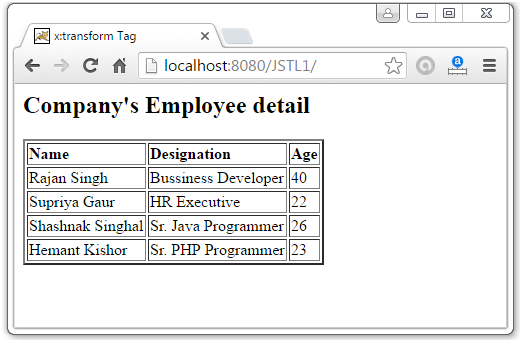
Now put the following program in **transfer.xml**, keeping in the same directory:

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<organisation>**
3. **<company>**
4. **<emp>**
5. **<name>**Rajan Singh**</name>**
6. **<designation>**Bussiness Developer**</designation>**
7. **<age>**40**</age>**
8. **</emp>**
10. **<emp>**
11. **<name>**Supriya Gaur**</name>**
12. **<designation>**HR Executive**</designation>**
13. **<age>**22**</age>**
14. **</emp>**
15. **</company>**
17. **<company>**
18. **<emp>**
19. **<name>**Shashnak Singhal**</name>**
20. **<designation>**Sr. Java Programmer**</designation>**
21. **<age>**26**</age>**
22. **</emp>**
24. **<emp>**
25. **<name>**Hemant Kishor**</name>**
26. **<designation>**Sr. PHP Programmer**</designation>**
27. **<age>**23**</age>**
28. **</emp></company></organisation>**

Now put the following program in **index.jsp**, keeping in the same directory:

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
3. **<html>**
4. **<head>**
5. **<title>**x:transform Tag**</title>**
6. **</head>**
7. **</html>**
8. **<c:import** var="xml" url="transfer.xml" **/>**
9. **<c:import** var="xsl" url="transfer.xsl" **/>**
10. **<x:transform** xml="${xml}" xslt="${xsl}" **/>**

Output:



# <x:param> Tag

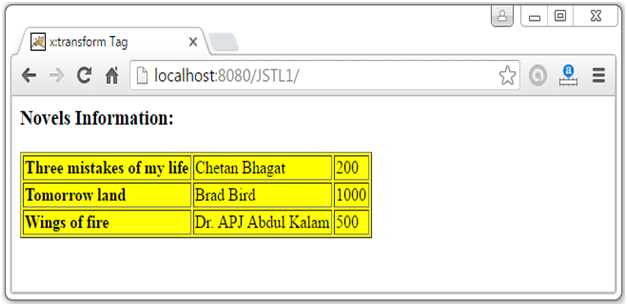
Let us put the following program in **transfer.xsl** file:

1. **<?xml** version="1.0"**?>**
2. **<xsl:stylesheet** xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"**>**
3. **<xsl:output** method="html" indent="yes"**/>**
4. **<xsl:param** name="bgColor"**/>**
6. **<xsl:template** match="/"**>**
7. **<html>**
8. **<body>**
9. **<xsl:apply-templates/>**
10. **</body>**
11. **</html>**
12. **</xsl:template>**
14. **<xsl:template** match="books"**>**
15. **<table** border="1" width="60%" bgColor="{$bgColor}"**>**
16. **<xsl:for-each** select="book"**>**
17. **<tr>**
18. **<td>**
19. **<b><xsl:value-of** select="name"**/></b>**
20. **</td>**
21. **<td>**
22. **<xsl:value-of** select="author"**/>**
23. **</td>**
24. **<td>**
25. **<xsl:value-of** select="price"**/>**
26. **</td>**
27. **</tr>**
28. **</xsl:for-each>**
29. **</table>**
30. **</xsl:template>**
31. **</xsl:stylesheet>**

Now put the following program in **index.jsp**, keeping in the same directory:

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %**>**
4. **<html>**
5. **<head>**
6. **<title>**x:transform Tag**</title>**
7. **</head>**
8. **<body>**
9. **<h3>**Novels Information:**</h3>**
10. **<c:set** var="xmltext"**>**
11. **<books>**
12. **<book>**
13. **<name>**Three mistakes of my life**</name>**
14. **<author>**Chetan Bhagat**</author>**
15. **<price>**200**</price>**
16. **</book>**
17. **<book>**
18. **<name>**Tomorrow land**</name>**
19. **<author>**Brad Bird**</author>**
20. **<price>**1000**</price>**
21. **</book>**
22. **<book>**
23. **<name>**Wings of fire**</name>**
24. **<author>**Dr. APJ Abdul Kalam**</author>**
25. **<price>**500**</price>**
26. **</book>**
27. **</books>**
28. **</c:set>**
30. **<c:import** url="transfer.xsl" var="xslt"**/>**
31. **<x:transform** xml="${xmltext}" xslt="${xslt}"**>**
32. **<x:param** name="bgColor" value="yellow"**/>**
33. **</x:transform>**
35. **</body>**
36. **</html>**

Output:



# JSTL SQL Tags

|  |  |
| --- | --- |
| **SQL Tags** | **Descriptions** |
| [sql:setDataSource](https://www.javatpoint.com/jstl-sql-setdatasource-tag) | It is used for creating a simple data source suitable only for prototyping. |
| [sql:query](https://www.javatpoint.com/jstl-sql-query-tag) | It is used for executing the SQL query defined in its sql attribute or the body. |
| [sql:update](https://www.javatpoint.com/jstl-sql-update-tag) | It is used for executing the SQL update defined in its sql attribute or in the tag body. |
| [sql:param](https://www.javatpoint.com/stl-sql-param-tag) | It is used for sets the parameter in an SQL statement to the specified value. |
| [sql:dateParam](https://www.javatpoint.com/jstl-sql-dateparam-tag) | It is used for sets the parameter in an SQL statement to a specified java.util.Date value. |
| [sql:transaction](https://www.javatpoint.com/jstl-sql-transaction-tag) | It is used to provide the nested database action with a common connection. |

# <sql:setDataSource> Tag

The <sql:setDataSource> tag is used for creating a simple data source suitable only for prototyping.

It is used to create the data source variable directly from JSP and it is stored inside a scoped variable. It can be used as input for other database actions.

**Example:**

Consider the below information about your MySQL database setup:

* We are using the **JDBC MySQL driver**
* We are using the **test**database on local machine
* We are using the **"root"** as username and **"1234"** as password to access the test database.
* **<**%@ 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>**sql:setDataSource Tag**</title>**
* **</head>**
* **<body>**
* **<sql:setDataSource** var="db" driver="com.mysql.jdbc.Driver"
* url="jdbc:mysql://localhost/test"
* user="root"  password="1234"**/>**
* **</body>**
* **</html>**