**JSTL functions and Core Tags- JSTL(JSP Standard Tag Lib)**

JSTL stands for JSP standard tag Library which is a collection of very useful core tags and functions. These tags and functions will help you write JSP code efficiently.

## JSTL Core Tags

Below is the collection of **JSTL core tags**. Each is explained with the help of proper examples. The following line of statement must be present in your JSP in order to use the JSTL core Tags.

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

1. <c:out> tag: It is used for displaying the content on client after escaping XML and HTML markup tags. Main attributes are default and escapeXML.
2. <c:set> tag: This tag is useful for setting up a variable value in a specified scope. It basically evaluates an expression and sets the result in given variable.
3. <c:remove> tag: It is used for removing an attribute from a specified scope or from all scopes (page, request, session and application). By default removes from all.
4. <c: if> tag: This JSTL core tag is used for testing conditions. There are two other optional attributes for this tag which are var and scope, test is mandatory.
5. <c:choose> tag: It’s like switch statement in Java.
6. <c:when> tag: It’s like case statement in Java.
7. <c:otherwise> tag: It works like default attribute in switch-case statements.
8. <c:catch>tag: This tag is used in exception handling. In this post we have discussed exception handling using <c:catch> core tag.
9. <c:import> tag: This JSTL core tag is used for importing the content from another file/page to the current JSP page. Attributes – var, URL and scope.
10. <c:forEach> tag: This tag in JSTL is used for executing the same set of statements for a finite number of times.
11. <c:forTokens> tag: It is used for iteration but it only works with delimiter.
12. <c:param> tag: This JSTL tag is mostly used with <c:url> and <c:redirect> tags. It adds parameter and their values to the output of these tags.
13. <c:url> tag: It is used for url formatting or url encoding. It converts a relative url into a application context’s url. Optional attributes var, context and scope.
14. <c:redirect> tag: It is used for redirecting the current page to another URL, provide the relative address in the URL attribute of this tag and the page will be redirected to the url.

## JSTL Functions

Following are useful **JSTL functions** with examples. Following Taglib directive should be included in the JSP page in order to use the JSTL functions.

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

1. fn:contains function: This function checks whether the given string is present in the input as sub-string. It does a case sensitive check.
2. fn:containsIgnoreCase(): It does a case insensitive check to see whether the provided string is a sub-string of input.
3. fn:indexOf(): It is used for finding out the start position of a string in the provided string. Function returns -1 when string is not found in the input.
4. fn:escapeXML(): It is used for HTML/XML character escaping which means it treats html/xml tags as a string. Similar to the escapeXml attribute of <c:out> tag.
5. fn:join() and fn:split() functions: JSTL functions: fn:join() concatenates the strings with a given separator and returns the output string. fn:split() splits a given string into an array of substrings.
6. fn:length(): The JSTL function fn:length() is used for computing the length of a string or to find out the number of elements in a collection. It returns the length of the object.
7. fn:startsWith(): It checks the specified string is a prefix of given string.
8. fn:endsWith(): fn:endsWith() JSTL function is used for checking the suffix of a string. It checks whether the given string ends with a particular string.
9. fn:substring(): This JSTL function is used for getting a substring from the provided string.
10. fn:substringAfter(): It is used for getting a substring which is present in the input string before a specified string.
11. fn:substringBefore(): It gets a substring from input which comes after a specified string.
12. fn:trim(): JSTL Function fn:trim() removes spaces from beginning and end of a string and function.
13. fn:toUpperCase(): It is just opposite of fn:toLowerCase() function. It converts input string to a uppercase string.
14. fn:toLowerCase(): This function is used for converting an input string to a lower case string.
15. fn:replace(): fn:replace() function search for a string in the input and replace it with the provided string. It does case sensitive processing.

# JSTL <c:out> Core Tag

**<c:out> is a JSTL core tag,**which is used for displaying server-side variables and hardcoded values on the browser (client). You may be wondering that a variable’s value and data can be displayed using [Expression language(EL)](https://beginnersbook.com/2013/11/jsp-expression-language-el/)  and [out implicit object](https://beginnersbook.com/2013/11/jsp-implicit-object-out-with-examples/) too then why do we need <c:out> jstl tag? the difference is that the <c:out> tag escapes HTML/XML tags but others don’t, refer the example to understand this.

### Tag <c:out> Example

In this example we are displaying a string on the browser, however we are using html tags in the value and we want to see what would be the result and how it is gonna HTML tags.

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

<html>

<head>

<title>c:out Tag Example</title>

</head>

<body>

<c:out value="${'<b>This is a <c:out> example </b>'}"/>

</body>

</html>

**escapeXml attribute of <c:out> tag**

Let’s say I modify the above code like this – I have just added escapeXML attribute in the tag and marked it false. **By default the value of escapeXML attribute is true**. Since we have marked it as false it would not escape HTML/XML tags and the tags will work.

<c:out value="${'<b>This is a <c:out> example </b>'}" escapeXml="false"/>

**Attribute “default” of <c:out> tag**

Above we have seen escapeXML attribute of the <c:out> tag. There is another attribute “default” for this tag, which is used to display the fallback or default value in case the value of the <c:out> tag is null. Here is the example where we are trying to print the value of string str using the tag and since the string str value is null, the tag is printing the value set in **default attribute**.

<%! String str = null; %>

<c:out value="${str}" default="default value of c:out"/>

# JSTL <c:set> Core Tag

**c:set> core JSTL tag** is used for assigning a value to an object or variable within a specified scope. Let’s understand this with an **example**.

Here I’m assigning a string value to a variable name within application scope (it will let me access my variable in any of the JSP page across application). On the other page (display.jsp) I have printed the value on browser using <c:out> tag and [EL](https://beginnersbook.com/2013/11/jsp-expression-language-el/).

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

<html>

<head>

<title>Example of c:set tag</title>

</head>

<body>

<c:set var="name" scope="application" value="Vishnu Pratap Singh"/>

<a href="display.jsp">Display</a>

</body>

</html>

display.jsp

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

<c:out value="${name}"/>

**Attributes of <c:set> tag**

**1) value:** It can be a hardcoded value or an expression. for e.g. below are the allowed variations of <c:set> tag –

The value of the variable myvar would be stored in the object name.

<c:set var="name" scope="application" value="${myvar}"/>

The result of the expression would be stored in the object.

<c:set var="sum" scope="application" value="${1+3+6}"/>

2) **var:** It holds the variable/object name

3) **scope:** It can be [request](https://beginnersbook.com/2013/11/jsp-implicit-object-request-with-examples/), [session](https://beginnersbook.com/2013/11/jsp-implicit-object-session-with-examples/), [page](https://beginnersbook.com/2013/11/jsp-implicit-objects/) and [application](https://beginnersbook.com/2013/11/jsp-implicit-object-application-with-examples/). In the above example we have specified the scope as application, however it can be anything out of the mentioned four. It all depends on the requirements.

# JSTL <c:remove> Core Tag

**<c:remove>** tag is used for removing an attribute from a specified scope or from all scopes (page, request, session and application).

## Example

In the below example, first I have set two variables using [<c:set> tag](https://beginnersbook.com/2013/11/jstl-cset-core-tag/) and then I have removed one of them using <c:remove> tag. As you can see in the output screenshot – when I tried to display both the variables, for the second attribute the page didn’t get any value and printed the default value using [default attribute of <c:out> tag](https://beginnersbook.com/2013/11/jstl-cout-core-tag/).

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

<html>

<head>

<title>Example of c:remove tag</title>

</head>

<body>

<c:set var="Site" scope="session" value="Facebook.com"/>

<c:set var="author" scope="session" value="Vishnu "/>

<c:remove var="author"/>

<a href="display.jsp">check attributes</a>

</body>

</html>

display.jsp

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

<c:out value="${Site}"/><br>

<c:out value="${author}" default="Attribute has no value"/>

## Scope attribute of <c:remove> tag

Above we have coded like this

<c:remove var="author"/>

This above code removes an attribute from all the scopes (page, session, application, request). In order to be specific we must need to specify the scope attribute inside <c:remove>  tag, like I did below – The below JSTL statement will remove the variable var from [session scope](https://beginnersbook.com/2013/11/jsp-implicit-object-session-with-examples/).

<c:remove var="author" scope="session"/>

# JSTL <c:if> Core Tag

**<c:if> is a JSTL core tag** which is used for testing conditions. It is more or like a if statement in java which evaluates a condition and executes a block of code if the result is true.

**Syntax:**

This is the basic syntax of <c:if> core tag. The set of statements enclosed within <c:if> tag gets executed if **test=”true”**. For using this tag we generally use [expression language](https://beginnersbook.com/2013/11/jsp-expression-language-el/) to evaluate an relational expression. We use EL because it returns boolean value(true/false) after evaluating the condition and we need the boolean value for test attribute.

<c:if test="${condition}">

...

..

</c:if>

## Example of <c:if> tag

In this example we have defined **age** variable using [<c:set> tag](https://beginnersbook.com/2013/11/jstl-cset-core-tag/) and then we are checking the eligibility of voting by using **<c:if> tag**.

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

<html>

<head>

<title>JSTL c:if Tag Example</title>

</head>

<body>

<c:set var="age" value="26"/>

<c:if test="${age >= 18}">

<c:out value="You are eligible for voting!"/>

</c:if>

<c:if test="${age < 18}">

<c:out value="You are not eligible for voting!"/>

</c:if>

</body>

</html>

## <c:if> attributes

Above we have seen the basic usage of <c:if> where we have used only the **test attribute**. However there are two other optional attributes for this tag which are **var** and **scope**. Using these attributes you can simply store the test results in a variable within a specified scope.

* var: Variable name in which the test result would be stored.
* scope: It defines the scope for storing the value. For e.g. if its session the stored var value can be accessed till the session is active.

## An example of var and scope attribute

Storing the test result in variable res in request scope. For printing the value we have given requestScope.res as the variable is stored in request however you can even give variable name(res) alone, it would work fine.

<c:if test="${17 >= 18}" var="res" scope="request">

</c:if>

<c:out value="${requestScope.res}"/>

# JSTL <c:choose>, <c:when>, <c:otherwise> Core Tags

In this we are discussing **<c:choose>, <c:when> and <c:otherwise> core tags** of JSTL. These tags are used together like **switch-case and default** statements in java. <c:choose> is the one which acts like switch, <c:when> like case which can be used multiple times inside <c:choose> for evaluating different-2 conditions. <c:otherwise> is similar to default statement which works when all the <c:when> statements holds false.

**Syntax:**

The basic structure looks like this –

<c:choose>

<c:when test="${condition1}">

//do something if condition1 is true

</c:when>

<c:when test="${condition2}">

//do something if condition2 is true

</c:when>

<c:otherwise>

//Statements which gets executed when all <c:when> tests are false.

</c:otherwise>

</c:choose>

## Example

In this example we have three numbers and we are comparing them using these three core tags. Example is pretty simple to understand.

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

<html>

<head>

<title>c:choose, c:when and c:otherwise Tag Example</title>

</head>

<body>

<c:set var="number1" value="${222}"/>

<c:set var="number2" value="${12}"/>

<c:set var="number3" value="${10}"/>

<c:choose>

<c:when test="${number1 < number2}">

${"number1 is less than number2"}

</c:when>

<c:when test="${number1 <= number3}">

${"number1 is less than equal to number2"}

</c:when>

<c:otherwise>

${"number1 is largest number!"}

</c:otherwise>

</c:choose>

</body>

</html>

In this we are discussing **<c:choose>, <c:when> and <c:otherwise> core tags** of JSTL. These tags are used together like **switch-case and default** statements in java. <c:choose> is the one which acts like switch, <c:when> like case which can be used multiple times inside <c:choose> for evaluating different-2 conditions. <c:otherwise> is similar to default statement which works when all the <c:when> statements holds false.

**Syntax:**

The basic structure looks like this –

<c:choose>

<c:when test="${condition1}">

//do something if condition1 is true

</c:when>

<c:when test="${condition2}">

//do something if condition2 is true

</c:when>

<c:otherwise>

//Statements which gets executed when all <c:when> tests are false.

</c:otherwise>

</c:choose>

## Example

In this example we have three numbers and we are comparing them using these three core tags. Example is pretty simple to understand.

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

<html>

<head>

<title>c:choose, c:when and c:otherwise Tag Example</title>

</head>

<body>

<c:set var="number1" value="${222}"/>

<c:set var="number2" value="${12}"/>

<c:set var="number3" value="${10}"/>

<c:choose>

<c:when test="${number1 < number2}">

${"number1 is less than number2"}

</c:when>

<c:when test="${number1 <= number3}">

${"number1 is less than equal to number2"}

</c:when>

<c:otherwise>

${"number1 is largest number!"}

</c:otherwise>

</c:choose>

</body>

</html>

In this article we are discussing **<c:choose>, <c:when> and <c:otherwise> core tags** of JSTL. These tags are used together like **switch-case and default** statements in java. <c:choose> is the one which acts like switch, <c:when> like case which can be used multiple times inside <c:choose> for evaluating different-2 conditions. <c:otherwise> is similar to default statement which works when all the <c:when> statements holds false.

**Syntax:**

The basic structure looks like this –

<c:choose>

<c:when test="${condition1}">

//do something if condition1 is true

</c:when>

<c:when test="${condition2}">

//do something if condition2 is true

</c:when>

<c:otherwise>

//Statements which gets executed when all <c:when> tests are false.

</c:otherwise>

</c:choose>

## Example

In this example we have three numbers and we are comparing them using these three core tags. Example is pretty simple to understand.

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

<html>

<head>

<title>c:choose, c:when and c:otherwise Tag Example</title>

</head>

<body>

<c:set var="number1" value="${222}"/>

<c:set var="number2" value="${12}"/>

<c:set var="number3" value="${10}"/>

<c:choose>

<c:when test="${number1 < number2}">

${"number1 is less than number2"}

</c:when>

<c:when test="${number1 <= number3}">

${"number1 is less than equal to number2"}

</c:when>

<c:otherwise>

${"number1 is largest number!"}

</c:otherwise>

</c:choose>

</body>

</html>

# JSTL <c:catch> Core Tag

**<c:catch> JSTL tag** is used in exception handling. Earlier we shared how to do exception handling in JSP – the two ways. In this post we are gonna discuss exception handling using <c:catch> core tag.

#### Syntax:

<c:catch var ="variable\_name">

//Set of statements in which exception can occur

</c:catch>

**variable\_name** can be any variable in which exception message would be stored. If there is an exception occurs in the statements enclosed in <c:catch> then this variable contains the exception message. Let’s understand this with the help of an example.

#### Example

In this example we are intentionally throwing arithmetic exception by dividing an integer with zero and  then we are printing the **errormsg** variable (which contains the exception message) using Expression language (EL).

**Note**: If there is no exception in the block of statements in <c:catch> then the variable (in example it’s errormsg) should have null value. That’s the reason we are checking errormsg!=null before printing the variable’s value.

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

<html>

<head>

<title>JSTL c:catch Core Tag Example</title>

</head>

<body>

<%!

int num1=10;

int num2=0; %>

<c:catch var ="errormsg">

<% int res = num1/num2;

out.println(res);%>

</c:catch>

<c:if test = "${errormsg != null}">

<p>There has been an exception raised in the above

arithmetic operation. Please fix the error.

Exception is: ${errormsg}</p>

</c:if>

</body>

</html>

# JSTL <c:import> Core Tag

JSTL <c:import> tag is used for importing the content from another file/page to the current JSP page.

#### Syntax:

<c:import var="variable\_name" url="relative\_url"/>

Here **variable\_name** is a variable which stores the data imported from another url.  
**relative\_url** is the address of the file/page which needs to be imported.

#### Attributes of <c:import>

* **url**: It’s mandatory attribute and needs to be mentioned always.
* **var**: It is an optional attribute if this is not specified then the imported data will be printed on the current page. For e.g. the statement <c:import url=”/file.jsp” /> would print the data of file.jsp on the client (browser).
* **scope**: It is also optional. If we are using var attribute then scope can be used along with it to specify the scope of the data stored in the variable.

#### Example

This is an page which has some data. We will import this page in index.jsp page.

display.jsp

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

<c:out value="Vishnu "/>

<c:out value="Facebook.com" />

<c:out value="This is just a String" />

index.jsp

Here we are importing the data from display.jsp into a variable **mydata** and then we are displaying it on browser using <c:out> tag.

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

<html>

<head>

<title> JSTL c:import Tag Example</title>

</head>

<body>

<c:import var="mydata" url="/display.jsp"/>

<c:out value="${mydata}"/>

</body>

</html>

# JSTL <c:forEach> and <c:forTokens> Core Tags

<c:forEach> tag in JSTL is used for executing the same set of statements for a finite number of times. It’s similar to the for loop in java. This is basically used when we need to perform(execute) set of statements again and again for a specified number of times.

<c:forTokens> is also used for iteration but it only works with delimiter which means using this tag we can break the input data into multiple parts based on the delimiter. We will understand this with the help of an example in this post.

## <c:forEach> Tag

#### Syntax of <c:forEach>

<c:forEach var="counter\_variable\_name" begin="intial\_value" end="final\_limit">

//Block of statements

</c:forEach>

The below are the three main attributes of <c:forEach> tag.

**begin:** The initial counter value.  
**end:**The final limit till which the loop will execute  
**var**: Counter variable name

#### Example

In this example we are printing value of variable counter in loop using <c:forEach> tag. The loop is starting from value 1 (mentioned in **begin** attribute) and ending at value 10 (value of **end** attribute).

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

<html>

<head>

<title>Example c:forEach tag in JSTL</title>

</head>

<body>

<c:forEach var="counter" begin="1" end="10">

<c:out value="${counter}"/>

</c:forEach>

</body>

</html>

## <c:forTokens> tag

#### Syntax of <c:forEach>

<c:forTokens items="value(s)" delims="delimiter" var="variable\_name">

//Set of statements

</c:forTokens>

The below are the three main attributes of <c:forTokens> tag.

**items:** Set of data value(s).  
**delims:**The delimiter can have any value. It can be a number, string or special character.  
**var**: variable name which stores the sub strings.

#### Example

In this example we are splitting the strings into multiple substrings using delimter dot(‘.’).

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

<html>

<head>

<title>Example c:forTokens tag in JSTL</title>

</head>

<body>

<c:forTokens items="www.facebook.com" delims="." var="site">

<c:out value="${site}"/>

</c:forTokens>

</body>

</html>

# JSTL <c:param> Core Tag

**<c:param>** JSTL tag is mostly used with <c:url> and <c:redirect> tags. Basically it adds parameter and their values to the output of these tags. In this tutorial we will see how the <c:param> tag can be used with  <c:url> and <c: redirect> tags.

#### Syntax:

<c:param name="parameter\_name" value="parameter\_value"/>

#### Attributes of <c:param> tag

* **name:** To specify the name of the parameter.
* **value**: To specify the value of the parameter.

#### Example of <c:param>

In this example we are using <c:param> tag for adding parameters to the resultant URL.

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

<html>

<head>

<title>JSTL c:param Tag Example</title>

</head>

<body>

<c:url value="/mypage.jsp" var="completeURL">

<c:param name="Id" value="736"/>

<c:param name="user" value="vishnu"/>

</c:url>

${completeURL}

</body>

</html>

#### Example 2: <c:param> use in <c:redirect> tag

Here we are passing parameters along with the redirect url using <c:param> tag and then we are displaying those parameters on the redirected page using param variable of expression language.

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

<html>

<head>

<title>c:param Example2</title>

</head>

<body>

<c:redirect url="/display.jsp" >

<c:param name="UserId" value="222"/>

<c:param name="UserName" value="Vishnu"/>

</c:redirect>

${completeURL}

</body>

</html>

display.jsp

USER ID IS: ${param.UserId}

USER NAME IS: ${param.UserName}

# JSTL <c:url> Core Tag

**<c:url> JSTL tag** is used for url formatting or you can say url encoding. This is mainly used when we need to open a JSP page based on the user input or based on the value of a variable. It basically converts a relative url into a application context’s url. It may sound confusing now but follow the given examples in this tutorial and you will be able to grasp it quite easy.

#### Syntax:

Basic syntax looks like this – The attribute “value” is a required attribute for the <c:url> tag

<c:url value="/file1.jsp" />

There are three other **optional attributes** exist for this tag which are as follows –

* **var**: Variable name to store the formatted url (resultant url).
* **context**: Used for specifying the application (or project name). Don’t get it? We will see this with the help of an example later.
* **scope**: The scope in which the var attribute would be stored. It can be request, page, application or session.

Let’s understand the use of this tag and attributes with the help of an example –

#### Example 1: value attribute of <c:url>

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

<html>

<head>

<title>JSTL c:url Tag Example</title>

</head>

<body>

<c:url value="/file1.jsp"/>

</body>

</html>

# JSTL <c:redirect> Core Tag

<c:redirect> is used for redirecting the current page to another URL.

#### Syntax:

<c:redirect url="http://www.anydomainhere.com/samplepage.jsp"/>

This is how the <c:redirect> tag looks like. We just need to provide the relative address in the **URL attribute** of this tag and the page will automatically be redirected the URL provided when it gets loaded.

#### Example

Here we are redirecting the page to a different url based on the value of the variable myurl. If the value is 1 page will be redirected to http://www.facebook.com and for 2 it will go to http://www.google.com.

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

<html>

<head>

<title> JSTL c:redirect Tag Example</title>

</head>

<body>

<c:set var="myurl" value="2" scope="request"/>

<c:if test="${myurl<1}">

<c:redirect url="http://www.facebook.com"/>

</c:if>

<c:if test="${myurl>1}">

<c:redirect url="http://www.google.com"/>

</c:if>

</body>

</html>